uc3m | Universidad Carlos III de Madrid

Doctoral Thesis

Control and Communication Systems for Automated Vehicles Cooperation and Coordination

Author:
Ahmed Hussein

Supervisors:
Prof. Dr. José María Armingol
Dr. Fernando García

Electrical Engineering, Electronics and Automation

Leganés, May, 2018

uc3m | Universidad Carlos III de Madrid

Doctoral Thesis

Control and Communication Systems for Automated Vehicles Cooperation and Coordination

Author: ***Ahmed Hussein***

Supervisor: ***Prof. Dr. José María Armingol***

Co-Supervisor: ***Dr. Fernando García***

Signatures of the examination court:

President:

Spokesperson:

Secretary:

Leganés, May, 2018

This work is dedicated to my family, who has always loved me unconditionally and has been constant source of support and encouragement to work hard for the what I aspire to achieve.
Thank you!

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this thesis are my original work toward the degree of Doctor of Philosophy at Universidad Carlos III de Madrid. This thesis is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text.

Ahmed Hussein
Leganés, May 2018

Acknowledgements

First and foremost, I thank God Almighty for giving me the strength, knowledge, ability and opportunity to undertake this thesis work and to persevere and complete it satisfactorily. Without his blessings, this achievement would not have been possible. I am grateful to all of those with whom I have had the pleasure to work during this project. Therefore, I would like to start by thanking my fellow colleagues for their feedback, cooperation and of course friendship. Furthermore, I would like to express my very great appreciation to my supervisors for their valuable and constructive suggestions during the planning and development of this thesis. Their willingness to give me time so generously has been very much appreciated. So thank you for your patient guidance, enthusiastic encouragement and useful critiques during this journey. Finally, thanks to each and everyone who was in one way or another contributed in the completion of this thesis.

Abstract

The technological advances in the Intelligent Transportation Systems (ITS) are exponentially improving over the last century. The objective is to provide intelligent and innovative services for the different modes of transportation, towards a better, safer, coordinated and smarter transport networks. The Intelligent Transportation Systems (ITS) focus is divided into two main categories; the first is to improve existing components of the transport networks, while the second is to develop intelligent vehicles which facilitate the transportation process. Different research efforts have been exerted to tackle various aspects in the fields of the automated vehicles. Accordingly, this thesis is addressing the problem of multiple automated vehicles cooperation and coordination. At first, 3DCoAutoSim driving simulator was developed in Unity game engine and connected to Robot Operating System (ROS) framework and Simulation of Urban Mobility (SUMO). 3DCoAutoSim is an abbreviation for "3D Simulator for Cooperative Advanced Driver Assistance Systems (ADAS) and Automated Vehicles Simulator". 3DCoAutoSim was tested under different circumstances and conditions, afterward, it was validated through carrying-out several controlled experiments and compare the results against their counter reality experiments. The obtained results showed the efficiency of the simulator to handle different situations, emulating real world vehicles. Next is the development of the iCab platforms, which is an abbreviation for "Intelligent Campus Automobile". The platforms are two electric golf-carts that were modified mechanically, electronically and electrically towards the goal of automated driving. Each iCab was equipped with several on-board embedded computers, perception sensors and auxiliary devices, in order to execute the necessary actions for self-driving. Moreover, the platforms are capable of several Vehicle-to-Everything (V2X) communication schemes, applying three layers of control, utilizing cooperation architecture for platooning, executing localization systems, mapping systems, perception systems, and finally several planning systems. Hundreds of experiments were carried-out for the validation of each system in the iCab platform. Results proved the functionality of the platform to self-drive from one point to another with minimal human intervention.

Resumen

Los avances tecnológicos en Sistemas Inteligentes de Transporte (ITS) han crecido de forma exponencial durante el último siglo. El objetivo de estos avances es el de proveer de sistemas innovadores e inteligentes para ser aplicados a los diferentes medios de transporte, con el fin de conseguir un transporte mas eficiente, seguro, coordinado e inteligente. El foco de los ITS se divide principalmente en dos categorías; la primera es la mejora de los componentes ya existentes en las redes de transporte, mientras que la segunda es la de desarrollar vehículos inteligentes que hagan más fácil y eficiente el transporte. Diferentes esfuerzos de investigación se han llevado a cabo con el fin de solucionar los numerosos aspectos asociados con la conducción autónoma. Esta tesis propone una solución para la cooperación y coordinación de múltiples vehículos. Para ello, en primer lugar se desarrolló un simulador (3DCoAutoSim) de conducción basado en el motor de juegos Unity, conectado al framework Robot Operating System (ROS) y al simulador Simulation of Urban Mobility (SUMO). 3DCoAutoSim ha sido probado en diferentes condiciones y circunstancias, para posteriormente validarlo con resultados a través de varios experimentos reales controlados. Los resultados obtenidos mostraron la eficiencia del simulador para manejar diferentes situaciones, emulando los vehículos en el mundo real. En segundo lugar, se desarrolló la plataforma de investigación Intelligent Campus Automobile (iCab), que consiste en dos carritos eléctricos de golf, que fueron modificados eléctrica, mecánica y electrónicamente para darle capacidades autónomas. Cada iCab se equipó con diferentes computadoras embebidas, sensores de percepción y unidades auxiliares, con la finalidad de transformarlos en vehículos autónomos. Además, se les han dado capacidad de comunicación multimodal (V2X), se les han aplicado tres capas de control, incorporando una arquitectura de cooperación para operación en modo tren, diferentes esquemas de localización, mapeado, percepción y planificación de rutas. Innumerables experimentos han sido realizados para validar cada uno de los diferentes sistemas incorporados. Los resultados prueban la funcionalidad de esta plataforma para realizar conducción autónoma y cooperativa con mínima intervención humana.

Table of contents

List of figures

List of tables

List of acronyms

ABS Anti-Locking Braking

ACC Adaptive Cruise Control

ADAS Advanced Driver Assistance Systems

CNP Contract Net Protocol

EKF Extended Kalman Filter

ESC Electronic Stability Control

GA Genetic Algorithm

GUI Graphical User Interface

I2V Infrastructure-to-Vehicle

iCab Intelligent Campus Automobile

ITS Intelligent Transportation Systems

LQR Linear Quadratic Regulator

MRS Multi-robot System

MRTA Multi-robot Task Allocation

mTSP multi-Travelling Salesman Problem

OSM Open Street Map

P2V Pedestrian-to-Vehicle

PID Proportional, Integral, and Derivative Controller

PWM Pulse Width Modulation

ROS Robot Operating System

RTT Round Trip Time

SA Simulated Annealing

SAE Society of Automotive Engineers

SLAM Simultaneous Localization and Mapping

SUMO Simulation of Urban Mobility

TCP Transmission Control Protocol

UAV Unmanned Aerial Vehicle

UGV Unmanned Ground Vehicle

UKF Unscented Kalman Filter

V2I Vehicle-to-Infrastructure

V2P Vehicle-to-Pedestrian

V2V Vehicle-to-Vehicle

V2X Vehicle-to-Everything

VANET Vehicle Ad-hoc Networks

VPN Virtual Private Network

VRU Vulnerable Road Users

WHO World Health Organization

List of symbols

Symbol	Description	Unit
a	Vehicle acceleration	m/s^2
δ	Ackermann reduced bicycle model steering angle	^{o}
δ_c	Ackermann column steering angle	^{o}
η	Motor efficiency	
F_b	Vehicle braking force	N
f_r	Vehicle rolling friction coefficient	m/s^2
g	Gravity of Earth, which is equals to 9.81	m/s^2
K_D	Derivative controller gain	
K_I	Integral controller gain	
K_P	Proportional controller gain	
K_t	Motor torque constant	$N.m/A$
L_m	Motor inductance constant	H
Q	The covariance matrix of the process noise	
r_1	Spur reduction ratio	
r_2	Worm gear reduction ratio	
R_m	Motor resistance constant	Ω
W_t	Vehicle total weight	kg

Chapter 1

Introduction

1.1 Motivation

It all started back in the 19th century, on the 3rd of July 1886, when Karl Friedrich Benz, a German mechanical engineer, introduced a horseless carriage to the public, the Benz Patent Motorwagen shown in Figure 1.1. It was the first practical automobile, it had three-wheels with a rear-mounted engine with no gears, which had a single-cylinder four-stroke with trembler coil ignition and reaching a top speed of 16 km/h in his first test-drive [1].

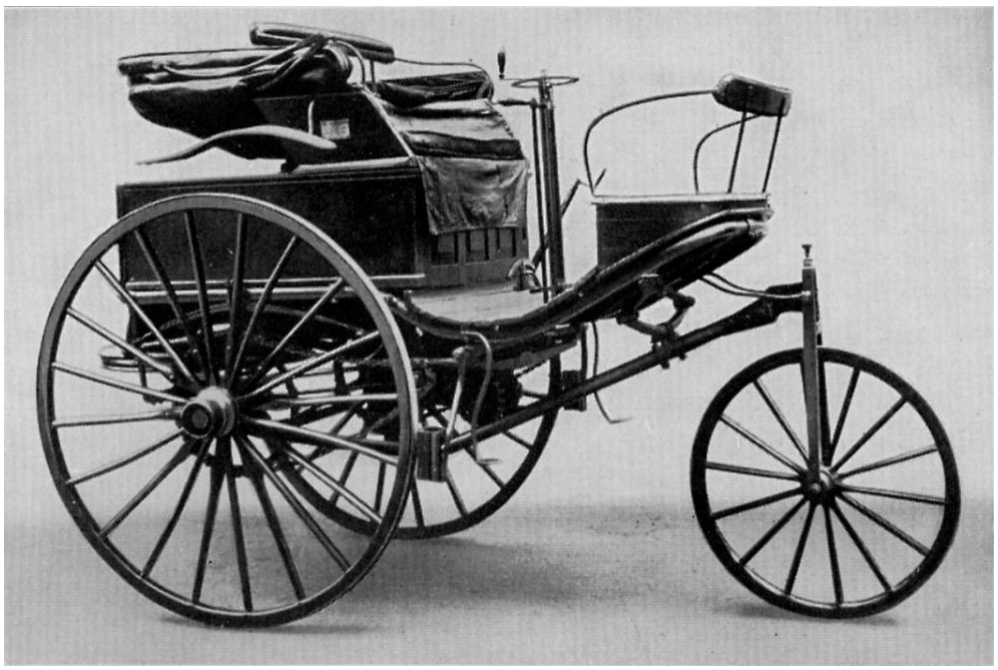


Figure 1.1 Benz Patent Motorwagen [1]

Shortly after during the demonstration, he crashed into a wall, due to the difficulty to control. Since then, governments and car manufacturers have been working around to reduce the error from the least reliable part of the vehicle, the driver. This is through improving three main aspects, the car strength, safety, and intelligence. For the first aspect, car strength, several types of research were conducted to study the car material, by reducing the car weight, using of high strength steel sheets, and introducing fiber-reinforced composites [2–6]. Fiber-reinforced polymers are one of the most common materials for the automotive industry, due to the fact that they offer enhanced characteristics, for instance, higher impact strength, easier to mold, better aesthetics, and lighter weight compared to the more common automotive components [7].

The second and most important aspect is the safety, in terms of both road and car safety. Accordingly, in May 2010, the United Nations has adopted the "Decade of Action for Road Safety" through the resolution A/RES/64/255, aiming to lower the predicted levels of road traffic casualties through five pillars; road safety administration; more secure roads and mobility; more secure vehicles; safer road users and improving after-crash response [8]. Accordingly, the following safety features were introduced[1]:

1. **Seat belts**: they were originated in the early 1900s, which developed to the standard three-point design in 1959. In the 1970s, seat belts became a standard in the cars.

2. **Shatter-resistant glass**: it was invented in the 1930s, using two layers of laminated, shatter-proof glass and a layer of plastic in between. Therefore, impact of a collision no longer break into shards.

3. **Airbags**: they were invented in the 1950s, starting with only driver airbags, then the side-impact airbags were included. In 1998, airbags became mandatory in all cars.

4. **Anti-Locking Braking (ABS)**: it was invented in the 1950s, which were actually first developed for aircraft to prevent wheels from locking when landing. In the 1980s, ABS became a standard in the cars production.

5. **Crumple zones**: it was invented in the 1950s, to absorb crash energy within the outer parts of the car rather than being transferred to passengers. Nowadays, all new cars are required to incorporate crumple zones for passenger protection.

6. **Electronic Stability Control (ESC)**: it was invented in the 1983, where it brakes individual wheels when needed and can cut engine power until control is regained. ESC became standard in car production in the early 1990s.

[1] Source: https://www.budgetdirect.com.au/blog/car-safety-features-in-modern-cars.html

The last aspect is the intelligence, which has the same purpose of improving the car safety, through the use of sensors. Figure 1.2 depicts an overview of a wide-range of on-board sensors. Accordingly, the ITS society introduces new technologies as part of the ADAS to reflect the increasing use of electronic and telecommunication technology within the road transport sector [9]. From the figure, it is noted that not all technologies are for car safety, but some can be for comfort, professional use or traffic management [10].

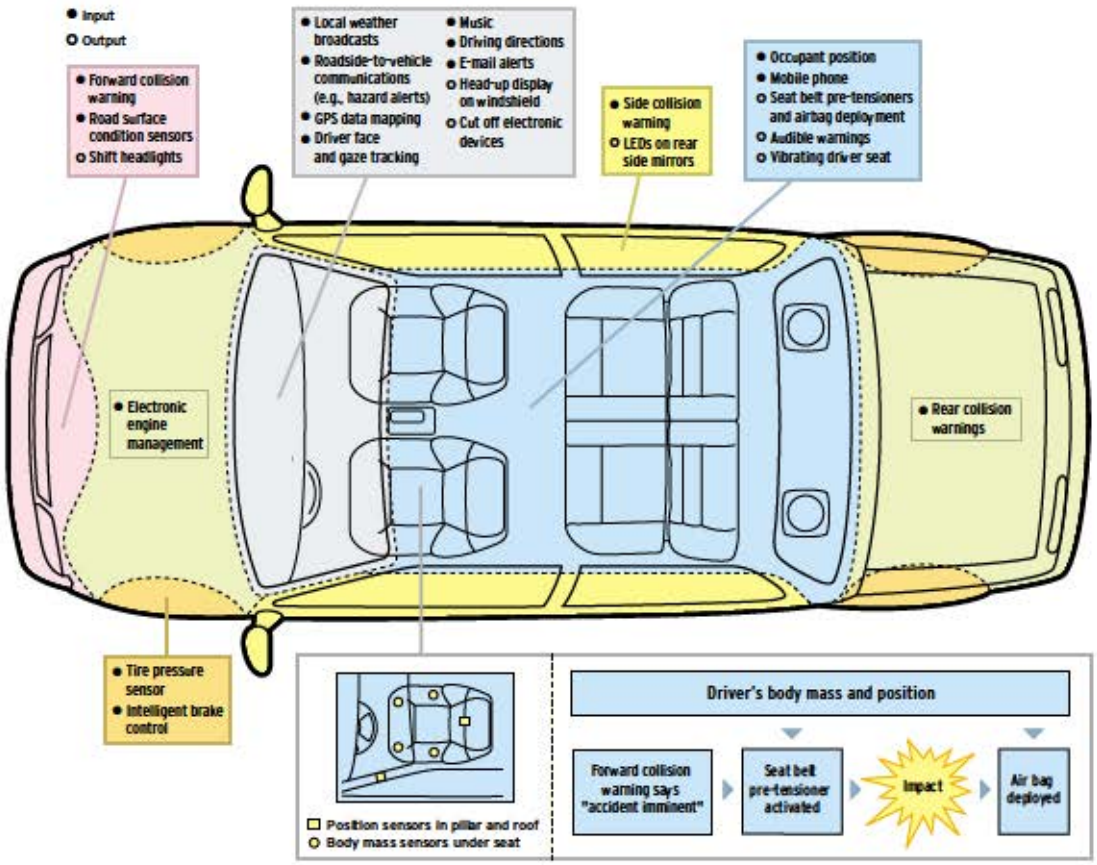


Figure 1.2 Sensors that can make cars safer [10]

In general, humans inherently have limited capacity for the number of simultaneously processed thoughts [11, 12]. Our lives are filled with distractions, and drivers take those distractions with them into the cars every day, which leads to distracted driving. Any activity that diverts attention from driving, including talking or texting, eating and drinking, fiddling with the navigation system, among others [13, 14]. According to the World Health Organization (WHO) reports, the traffic accidents lead to over 1.6 million crashes every year [15]. One out of every four car accidents is caused by the use of cell phones [16].

Accordingly, the term intelligent vehicle is introduced and Society of Automotive Engineers (SAE) proposed the five driving automation levels [17], presented in Figure 1.3. The first level is the assisted driving, where the driver is continuously exercising longitudinal control, while the system is performing the lateral control, for example, cruise control system [18]. The second level is the partial automation, where the driver has to monitor the systems at all times and be ready to take control, while the system has longitudinal and lateral control, for example, Adaptive Cruise Control (ACC) system [19] combined with lane keeping assist system [20]. These systems are currently functional in many cars for over a decade. The third level is the conditional automation, where the driver must be in a position always ready to take-over control, if needed, while the system has both longitudinal and lateral control and is aware of the surrounding environment during a defined use case, thus it recognizes when the driver is needed to take over, currently Tesla Autopilot car is the closest to reach this level [21]. The fourth and fifth levels are not yet reached and they are quite similar; on the one hand, in the high automation level, the driver is not required during the defined use case and the system can cope with all situations automatically in this use case. On the other hand, in the full automation, the driver is still not required and the system can cope with all situations automatically during the entire journey and in all cases. In other words, the car is responsible for accelerating, braking, steering, monitoring the car (self-awareness), perceive the surrounding environment and roadway; as well as responding to events, determining when to change lanes, turn, and use signals.

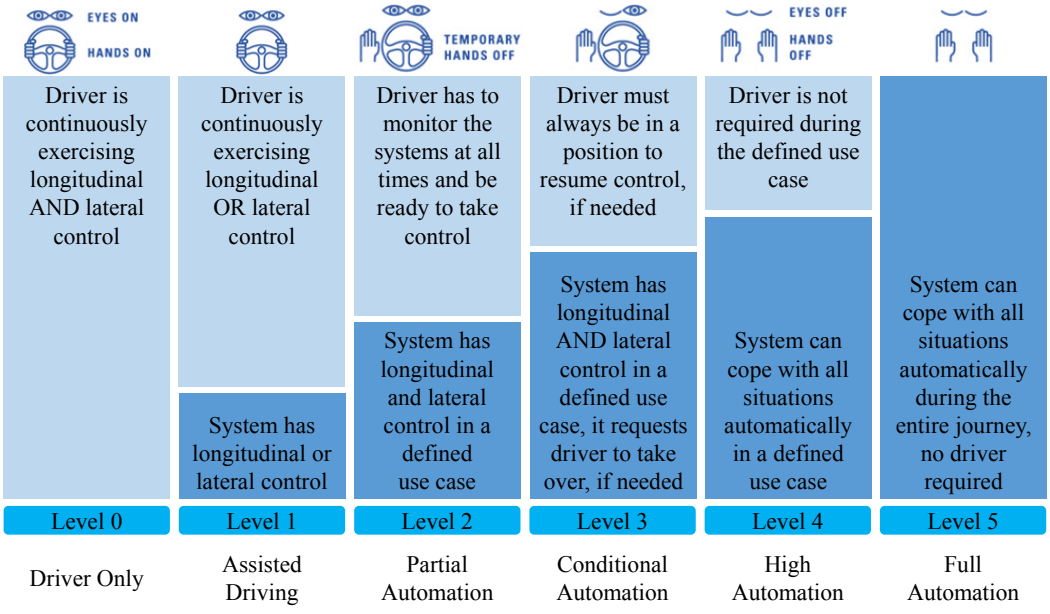


Figure 1.3 SAE 2016 five driving automation levels [22]

At the same time of the rapid technological development in the automotive industry over the past century, cities and populations are growing dramatically, with increasing transportation requests among others. In order to manage the whole system and optimize the outcome, the smart city is introduced and in consequence smart mobility is introduced [23]. From smart mobility to shared mobility [24], they are all leading to cooperative automated driving for intelligent vehicles [25].

In conclusion, the necessity of intelligent vehicle is crucial, to lower the number of accidents, traffic congestion, air pollution and improve quality of life. Additionally, the more intelligent vehicles are introduced to the roads, the more cooperation and coordination among them is required. Therefore, it is of a great importance to research, develop and propose an efficient approach to solve the multiple automated vehicles cooperation problem.

1.2 Objectives

Accordingly, this thesis is addressing the problem of multiple automated vehicles cooperation and coordination, by proposing an approach to solve it through various stages. First stage is the study of simulation environments, since it is essential to understand how the vehicle is going to behave in simulation before deployment in the real roads. Therefore, a powerful 3D simulator is designed for simulation of any type of ground vehicle with a various number of on-board devices. This simulator is tested under different circumstances and conditions, moreover, it is validated through carrying-out several controlled simulated experiments and compare the results against their counter reality experiments with multiple drivers.

The second stage is to use different platforms for testing the proposed approaches to automation. Therefore, small mobile robots were selected to carry out several indoor experiments of both cooperation and coordination algorithms. Moreover, an unmanned aerial vehicle platform was selected to test the proposed planning approach. The purpose of the carried-out experiments was the verification of the generic aspect of the proposed approaches and their functionality on heterogeneous platforms.

The third stage is the main focus of this thesis, which is the full development of a highly automated intelligent vehicle. Multiple electric golf carts were selected to be the vehicles, where they were modified electrically, mechanically and electronically to achieve conditional automation and reach one step closer to the high automation level. Moreover, multiple devices and sensors were installed on the platforms, to acquire data for the environmental analysis and understanding. For the self-driving goal, several approaches to localization, mapping, perception, and planning were implemented. In addition to multiple communication schemes for the cooperation and coordination approaches.

The fourth stage is to utilize the developed automated vehicles to analyze the effect of communication schemes in enhancing the environment perception under different conditions and constraints. Furthermore, a platooning approach was introduced as a use case for cooperative driving. The approach was analyzed using the automated vehicles and road users in an off-road urban environment through a number of experiments.

The fifth stage is to design an architecture for the multiple automated vehicle coordination, focusing on solving the shared mobility-on-demand transportation requests problem. The architecture is designed in a generic manner, to allow heterogeneity, scalability, adaptability, and integrability to various platforms. Moreover, the proposed coordination architecture took into consideration the connection to the vehicle software architecture to execute the assigned tasks in real-world experiments and not limited to only simulation experiments.

Finally, the last stage is to propose a solution method to the coordination problem, which is able to allocate the vehicles to the requests in real-time and near-optimal solution. The objective function is to optimize the distance covered by each vehicle, the overall execution time, the waiting time of the users, and the vehicle energy consumption. The proposed hybrid approach was compared to various benchmarks to extensively test its capabilities of handling tasks allocation.

In summary, the thesis main contribution is to present control and communication systems for multiple automated vehicles in order to solve the cooperation and coordination problems. The systems are tested in several simulations and real-world experiments for validation of the proposed work, ensuring its functionality and extendability to heterogeneous vehicles.

1.3 Structure

This thesis consists of eight chapters, including this one, and three appendices. Each chapter begins with a brief introduction for its contents and ends with a brief concluding remarks to the obtained outcomes. The majority of the presented work in this thesis is validated through various publications in different conferences, journals, and books. The structures of each element is summarized as follows:

- Chapter 1: introduces a general introduction to the thesis by shedding a light on the motivation of the work and the target objectives and contribution.
- Chapter 2: gives a general overview of the different simulation environments, and different automated vehicles. Followed by a review to the previous work in cooperative driving and task allocation problems.

- Chapter 3: presents the utilized softwares for the simulators, highlighting their advantages and disadvantages. In addition to, a detailed presentation for the designed driving simulator.
- Chapter 4: presents brief details to the secondary platforms of this thesis, the indoor mobile robot and the outdoor Unmanned Aerial Vehicle (UAV). Furthermore, it presents a detailed report to the main platform of this thesis, the Unmanned Ground Vehicle (UGV), highlighting the designed architecture, mounted devices, implemented technologies and functionalities.
- Chapter 5: demonstrates the proposed schemes for different vehicle communication approaches and their enhancements to the automated vehicles environment perception. Furthermore, the proposed platooning approach is presented as a use case of cooperative driving.
- Chapter 6: introduces the multiple vehicles coordination Robot Operating System (ROS)-based architecture for transportation requests. The architecture flowchart process is described, detailing the intelligibility with heterogeneous platforms. Furthermore, a hybrid optimization-based algorithm is presented, in order to solve the allocation problem of requests for the automated vehicles.
- Chapter 7: validates the proposed work and approaches by presenting obtained results from all carried-out experiments. Each section includes the experimental setup, the selected scenarios description, the selected evaluation metrics and finally a discussion of the results in qualitative and quantitative analysis.
- Chapter 8: summarizes the carried-out work and presents future recommendations.
- Appendix A: presents the publications list derived from the thesis carried-out work.
- Appendix B: shows the list of supervised theses during the framework of this thesis.
- Appendix C: introduces the Read-Me documentation of the implemented simulator.

Chapter 2

State-of-the-art

2.1 Introduction

This chapter presents a state-of-the-art for the related work of this thesis based on the proposed objectives. In Section 2.2, a review for the different simulation environments is presented, showing the attempts to link simulators with vehicle controllers framework, in order to reach a more accurate virtual emulation. Section 2.3 presents a brief history of various forms of automated vehicles hardware and software architecture. Afterward, the literature review for cooperative driving approaches is presented through several use-cases in Section 2.4. Additionally, Section 2.5 introduces the task allocation problem for vehicles, showing different approaches to solve the problem. Finally, the chapter is summarized in few concluding remarks.

2.2 Simulation Environments

Driving simulators have been used in many applications, including traffic safety, ADAS implementation, driver distraction, human-machine-interaction, among others. For example, to evaluate the usability of head-up displays in forwarding collision warning systems [26] or to assess a user interface for a novel traffic regulation system [27], in addition to several examples for simulation models are described in [28]. The authors in [29] combined the network simulator ns-2 [30] with the open source traffic simulator Simulation of Urban Mobility (SUMO) [31] to evaluate Vehicle Ad-hoc Networks (VANET) and developed a TraCI [29], in which SUMO and ns-2 communicated over a Transmission Control Protocol (TCP) connection to simulate inter-vehicular communication. In a further work, SUMO

was also integrated with the network simulator OMNeT++, in order to evaluate inter-vehicle communication protocol [32, 33].

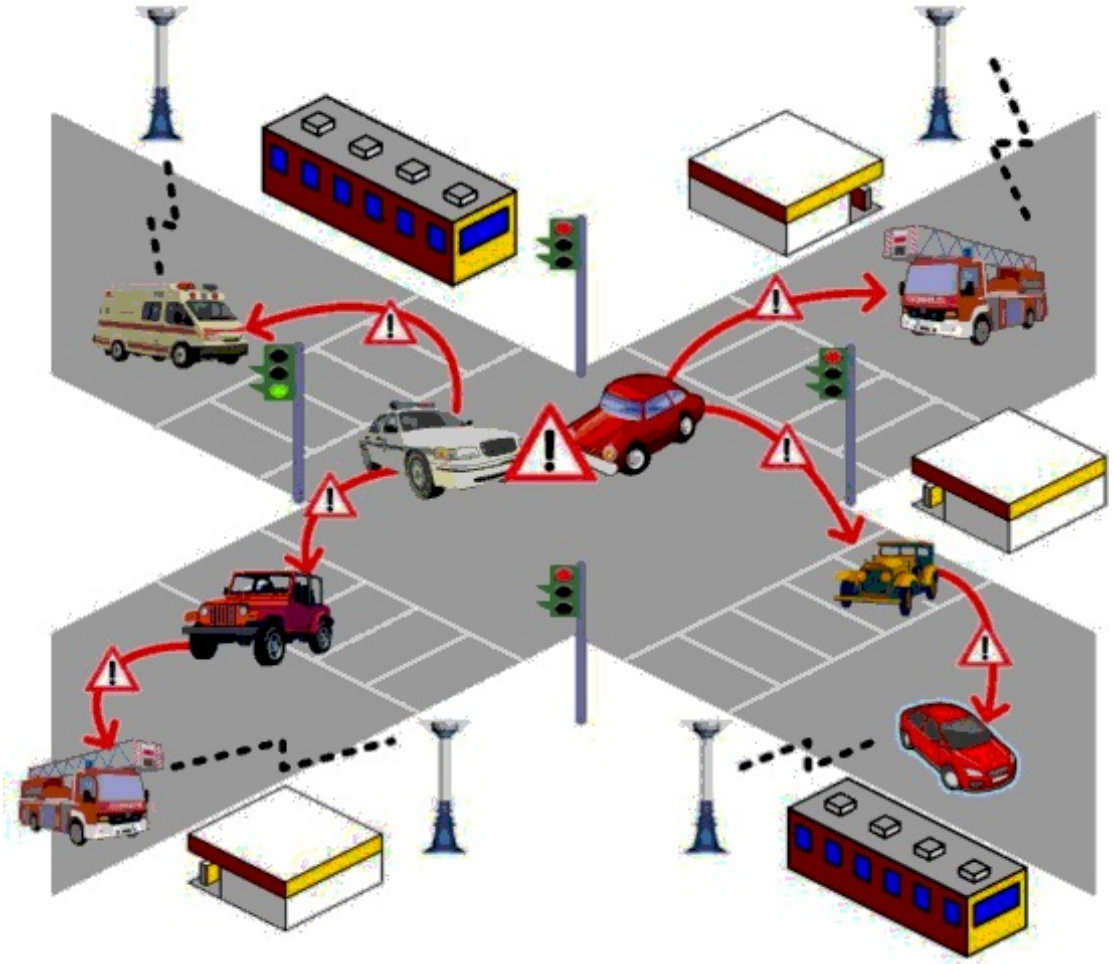


Figure 2.1 Network simulator ns-2[1]

Furthermore, the Udacity self-driving car nano degree program offers a simulator to teach students how to train cars and how to navigate road courses using deep learning [34], the authors utilized Unity game engine for the simulation environment. On the one hand, authors in [35], utilized Unreal engine to develop CARLA simulator, which has been developed to support training, and validation of self-driving urban systems. On the other hand, authors in [36] developed OpenDS driving simulation platform for various driving tasks and environment design.

[1]Source: https://ns2projects.org/ns2-simulation-code-for-vanet

Figure 2.2 Udacity self-driving car simulator [34]

Intelligent vehicles simulations are utilized as the initial step of experiments before the deployment on the roads. In the past, the possibility of having a simulator that emulates real-life was not straightforward, due to the lack of common software systems that relate the vehicle control to the virtual environment. Nowadays there are several frameworks that can be used to control vehicles, and ROS is the most common one [37]. Moreover, there are several powerful visualization tools that can be used for simulations, and Unity Game Engine is on the top of the list [38]. Accordingly, several attempts towards the link between ROS and Unity were made. In 2013, authors in [39] introduced Unity-Link protocol, which is a stream-oriented, layered and componentized design. The link was through Python scripts to send and receive data over TCP/IP sockets, however, a detailed analysis of the performance was not available, since it was out of their scope. Later in 2014, authors in [40] proposed a methodology to connect from ROS to Unity, to send data over TCP/IP sockets. This was the first attempt to utilize *rosbridge* library [41], which responsible for taking JSON [42] strings and converting them to ROS messages, and vice versa. They verified their approach through leveraging a mobile robot via virtual environment display and interaction in Unity.

On the other hand, in 2015, authors in [43] presented a 3D simulation system for miniature unmanned aerial vehicles. The exchange of data packets between the Unity server and the ROS client was achieved by TCP/IP protocol. On the client side, a ROS node was developed *unitysocket*, which communicates to the preset IP address of the Unity server. Authors

validated their approach through various experiments and proved the ability to glue ROS and Unity together for near real-time Unmanned Aerial Vehicle (UAV) simulations [44]. In 2017, authors in [45] presented several scripts which establish the connection and allows for invoking ROS services. The objective was to monitor and control industrial robotic arm and the approach was validated with satisfactory results.

Also in 2017, authors in [46] proposed a connection mechanism to bind ROS and Unity for virtual reality applications. The link was again through *rosbridge* library, utilizing both JSON and BSON strings and converting them to ROS messages. The results from this approach were validated using human-robot interaction in virtual reality simulator. The idea of *rosbridge* library to connect ROS and Unity obtained the best results, therefore Michael Jenkins created *unityros* library [47]. This Unity library was implemented to send and receive data of TurtleBot [48]. Then Thorstensen enhanced the library in [49] to include more messages and move one step closer to the generalization [50]. Accordingly, to sum up, a generic methodology to connect both systems, ROS, and Unity is lacking, which is addressed through work in Chapter 3.

2.3 Automated Vehicles

Automated vehicles could revolutionize how people get around. With the introduction of automation into roads, it will contribute to solving the issues related to the traffic accidents, congestion, and energy consumption. The self-driving vehicle technologies are advancing on a great scale, specifically the multiple sensors fusion techniques, deep learning and computational intelligence. Together, they enable these vehicles to understand the nearby surroundings and take appropriate actions to navigate on their own from one point to another [51–53].

The first two main demonstrations of the capabilities of self-driving vehicles took place in the United States through the Defense Advanced Research Projects Agency (DARPA) to develop automated cars. First one was in 2004 and 2005, the DARPA Grand Challenge had the races in the desert with no dynamic obstacles [54]. The second one was in 2007, the DARPA Urban Challenge had the race in an urban circuit among automated cars, simulating the dynamic traffic as in real urban environments [55], as shown in Figure 2.3.

Self-driving cars are becoming more frequent in both scientific and industrial context since Google launched their driverless car project in 2011 [56]. From then onwards, many other approaches proposed different architectures and solutions all of them moving towards the development of the autonomous vehicle. For instance, Mercedes with the Bertha project proved the viability of Autonomous vehicles in German roads based on advanced sensing

Figure 2.3 DARPA grand challenge cars[2]

capabilities [57]. The V-Charge project studies in the direction of allowing automated valet parking for self-driving cars [58]. Moreover, several vehicle manufacturers have proposed different solutions in the field of automated vehicles which are close to the market, such as BMW and Audi [59], Mercedes-Benz [60] and Volvo [61]. Furthermore, several other proposals offer the possibility to include autonomous vehicles in public transportation systems [62, 63]. Automated vehicles continue as an important topic in ITS, where a recent work shows that driver-less vehicles could become widely available in the next 5 to 10 years [64]. In this thesis, multiple automated vehicles were designed to examine the proposed approaches in real-life experiments. The development of the automated vehicles required the study of the various system, which is summarized in the below subsections.

[2]Source: https://medium.com/self-driving-cars/failure-and-the-darpa-grand-challenge-92c8dd63b25b

2.3.1 By Wire Systems

The by wire system is the first step to implement an automated vehicle. It can be defined as the replacement of mechanical linkages and systems with electrical or electro-mechanical ones for performing traditional vehicle functions [65]. The system consists of three categories; steer, drive, and brake by wire systems. On the one hand, the steer by wire system replaces the conventional steering system of the vehicle and aims to eliminate the physical connection between the steering wheel and the wheels causing high-speed stability [66].

On the other hand, the drive by wire system executes the vehicle propulsion by means of an electronic throttle without any cables from the accelerator pedal to the throttle valve of the engine. However, for electric vehicles, this system controls the electric motors connected to the wheels by sensing the accelerator pedal input and sending commands to the power interface modules [67]. Finally, the brake by wire system eliminates traditional mechanical and hydraulic components and replaces them with electronic sensors and actuators to control the brakes in vehicles [68]. In this thesis, all three by wire systems were developed on multiple automated vehicles, presented in Chapter 4, and later tested in several experiments as presented in Chapter 7.

2.3.2 Communication Systems

The vehicular communication technologies consist of three main categories; inter-vehicle communication, communication with pedestrians, and communication with infrastructure [69]. Vehicle-to-Everything (V2X) technologies augment vehicles on-board sensors and perception systems with non-line of sight awareness [70]. Each of the communication categories consists of several schemes, and several solutions utilize Ad-hoc networks, which are created by the temporarily connected vehicles. These networks are known as VANET [71]. In VANET, vehicles transmit data to each other when they are located in the defined range of connectivity from each other. Accordingly, VANET main drawback is its dependence on vehicle position in the environment, which does not guarantee constant connection.

Vehicle-to-Vehicle (V2V) communication scheme is defined as the transmission of data among multiple vehicles. The main objective of V2V communication is to enhance the perception capabilities of the vehicle, therefore reducing the number of traffic accidents [72]. Vehicle-to-Pedestrian (V2P) and Pedestrian-to-Vehicle (P2V) communication schemes enclose a wide range of road users, which allows bilateral detection and notification systems [73]. Last but not least, the systems can be enhanced through communicating with infrastructure. Vehicle-to-Infrastructure (V2I) and Infrastructure-to-Vehicle (I2V) communication schemes provide enhancements to the automated vehicles, for instance, the suggestion

of dynamic velocities or routes based on the instant information of traffic lights [74]. The use of this technology allows to the urban traffic coordination, ramp metering, monitoring and pollution management for control purposes in the smart cities [75]. The three categories of vehicular communication are designed and created within the framework of this thesis, as presented in Chapter 5.

2.3.3 Localization Systems

The localization system is considered as one of the important parts in any intelligent vehicle. The accuracy of the localization system must be as high as possible; in order to avoid any catastrophic maneuver. Sensors of high accuracy data are available for many decades; however, they are still unaffordable for mass production. Therefore, it is required to increase the accuracy of the localization using multiple cost-efficient sensors, which is achieved by the calibration, modeling, filtering and fusion processes. The good knowledge about the covariances and the identification of the uncertainties are fundamental; in order to obtain a filter close to optimal. Tuned and constant covariance parameters are widely used in different works [76, 77]; this is due to the simplicity of the calculations. However, it is not sufficient because the uncertainty in the drift suffering sensors changes with operating conditions, which leads to sub-optimal fusion results. Moreover, it needs a significant number of trails to obtain tolerable results.

Several methods have been presented to estimate the noise covariance matrices; such as correlation techniques that are based on Autocovariance Least-Square method (ALS) [78, 79], Covariance Matching (CM) [80, 81], maximum likelihood [82, 83], and Bayesian estimation [84, 85]. These previous methods usually estimate the covariances of linear systems, except CM methods which deal with nonlinear systems, however, it is a non-optimal estimator. Other works used Adaptive Kalman filters; in order to estimate the covariance matrices [86]. In [87], a Kalman filter with recursive estimation has been presented; to estimate the noise covariance matrix from the measurement sequence of linear time-invariant systems. Additionally, in [88], a stability analysis has been performed to verify the state of the estimator. However, the obtained results were based on simulations, furthermore, the estimation algorithm is efficient with linear systems.

Using filters requires accurate covariances for all the sensors being fused, however, determining the exact values of these covariances is difficult, due to the absence of a ground truth for the exact estimation. Moreover, the values of these covariances might change during the operation of the vehicle. Based on the literature, it is common to manual tune these covariances, which lead to the filters to operate sub-optimally and obtain worse results. Accordingly, in this thesis, an online adaptive estimation of the noise covariance matrix Q for

drift suffering proprioceptive sensors, using an exteroceptive sensor with known uncertainty is presented. This estimation is independent of the factors that affect the sensor reading, and it is not subjected to the accumulation of the error, but to the random white noise error. Furthermore, a comparative study was conducted using multiple odometry sensors fusion algorithms for validation of the approach efficiency and functionality.

2.3.4 Mapping Systems

The classification of map types has been changing over time, the most common categories are occupancy grid maps, feature maps, and topological maps [89]. Though topological maps are used in higher-levels of control, they do not contain as much information as occupancy grid maps, which are more common for the representation of the environment and the planning of trajectories [90]. Therefore, in order to achieve self-driving, the vehicle must be able to create a map of the world and localize itself in it. Several researchers addressed the environment mapping in [91, 92], furthermore, Simultaneous Localization and Mapping (SLAM) techniques, in particular, have been an active area of research [93, 94]. However, since the accuracy of the mapping systems must be as high as possible for better environment perception; it is required to increase the accuracy using multiple sensors, which is achieved by the calibration, modeling, filtering and fusion processes [95, 96].

2.3.5 Perception Systems

One of the necessary components to develop an automated vehicle is the perception of the surrounding vehicle environment. Perception systems are able to sense and interpret surrounding environment based on various kinds of sensors [97]. Several approaches for obstacle detection and classification are proposed; vision-based techniques [98, 99], lidar-based techinques [100, 101], and most recently deep learning based techniques [102, 103].

2.3.6 Planning Systems

Path planning has been a subject of study for the last decades. Most of the authors divide the problem into global and local planning. A review of the different approaches and concept definitions are presented in [104]. Depending on the analysis of the map, some methods are based on roadmaps, such as [105], others solve the problem by assigning a value to each region of the map in order to find the path with minimum cost [106]. A similar approach by using potential fields is described in [107], the most extended algorithm used a few years ago rapidly-exploring random trees [108] or the new approach based on neural networks [109].

2.4 Cooperative Driving

The self-driving technology is advancing rapidly, however, in order to safely deploy vehicles on public roads, cooperation with other road users is mandatory. This cooperation would allow the safe interaction with other vehicles, whether a human driver or driverless. Although some of the proposed solutions already handle the presence of other vehicles in the road [104, 110, 111], by handling static and dynamic obstacles and adapting the trajectory accordingly, the cooperative driving is not yet achieved.

Cooperative driving demands the information exchange and control strategies deployed in all the vehicles involved following a two-way communication. In this sense, during recent years many works have addressed this topic, trying to provide solutions based on different configuration and solutions. The Grand Cooperative Driving Challenge 2016 was created with the purpose to boost cooperative automated vehicles in the form of a cooperative based competition [112]. On the one hand, authors in [113] presented an auction-based cooperative control for autonomous vehicles, on the other hand, authors in [114] proposed an approach to enhance common motion planning algorithms, this proposal allows cooperation with human-driven vehicles. Additionally, a novel concept is presented, based on a centralized strategy, using maneuver templates, which are formalized collaborative maneuvers, to select cooperative driving strategies [115]. All this work prove the importance of collaboration and communication among vehicles to allow a safe and efficient deployment of autonomous vehicles real in driving scenarios.

Coordination of automated vehicles is part of the cooperative driving approaches. Since coordination ensures the safety of road users and enhances the efficiency of road traffic [116]. A use case of the coordination approaches is the vehicles platooning. Platooning is defined as multiple vehicles following each other in a coherent system, which has been widely used in several intelligent vehicles applications [117]. Furthermore, automated highway systems using platooning approaches have been demonstrated over the past several years [118–121]. Automated vehicles platooning can be classified into three main categories:

- **Longitudinal control**; where the follower vehicles are required to maintain a specific distance to the leader vehicles, in meantime maintaining the same driving speed.
- **Lateral control**; where the follower vehicles are required to execute lane tracking and lane changing algorithms to follow the leader vehicles on the road.
- **Maneuver coordination**; where the platoon formation plays an important role in joining and leaving protocols to assure smooth and safe driving on the road.

2.5 Task Allocation

During the last decade, Multi-robot System (MRS) is one of the main focus topics for the ITS field. This increased interest originates from the significant benefits provided by several robots over a single one. MRS can be simply understood to be a group of robots cooperating together for accomplishing a certain task or mission [122, 123]. In reference to the literature review survey, the coordination and cooperation among multi-agent systems can be modeled as a MRTA problem. MRTA is an NP-hard problem, which concerns the use of the available resources in an coherent manner. Consequently, the decision of which robot will do which task strongly affects the performance of the system [124, 125].

MRTA problem answers the question of which robot is going to execute which task? In the literature, there are different approaches utilized to solve the MRTA problem. Authors in [126] proposed a mixed integer linear programming optimization approach to allocate heterogeneous robots for maximizing the coverage area of the environment. In [127], authors proposed a SA approach to solve the MRTA through multi-Travelling Salesman Problem (mTSP) formulation. In [128], authors introduced the principles of a market economy to the coordination problem of multiple robots. In [129], authors proposed market-based approach over Contract Net Protocol (CNP) to solve MRTA problem. Moreover, the task allocation problem was also solved using hybrid optimization approaches such as the tabu search with random search method in [130] and tabu search with noising method in [131].

On the one hand, authors in [132] presented a centralized method for MRTA and path planning, and [133] authors simulated several planners based on centralization and decoupling. On the other hand, authors in [134] presented a formation control of MRS using decentralized approach, since the robots in the system lack the information about the environment and the positions and goals of other robots . And, in [135], authors presented a mobile sensor network using MRS, they used a decentralized approach in estimation and control of the agents to maintain connectivity during motion . Furthermore, in [136], ant-colony approach was proposed to solve MRTA problem for UAV to multiple targets in a military application scenario. As per the survey in [137], there are different approaches to solve the MRTA problem. Though several techniques are presented with satisfactory results, they lack the applicability and integrability to real automated vehicles, additionally, they lack online distributed computation on multiple vehicles [138]. Accordingly, in this thesis, a task allocation architecture for multiple automated vehicles is presented in Chapter 6. Moreover, the architecture was integrated into the developed automated platforms and tested for validation.

2.6 Concluding Remarks

Based on the conducted literature review, it can be concluded that the research of the automated vehicles has been diverse and extended over wide disciplines. However, despite this, there are few points that need more research, in order to reach the desired driving automation level. Since a generic methodology to connect both systems, ROS and Unity is lacking, a powerful 3D simulator development is proposed within the framework of the thesis, which is addressed through work presented in Chapter 3. In Chapter 4, multiple automated vehicles were designed to be utilized as the platforms for all experiments in this thesis, starting from small indoor mobile robots, to an automated UAV, and finally the main platforms of the thesis, which are two automated Unmanned Ground Vehicle (UGV). Moreover, Chapter 4 details all the developed intelligent systems in the platforms to reach the goal of automated driving. The developed communication schemes to enable the cooperative driving are explained in Chapter 5, along with the description of the proposed approach for multiple vehicles platooning. Finally, the coordination architecture among the multiple automated platforms is presented in Chapter 6, where the MRTA problem was solved using a hybrid optimization-based algorithm.

Chapter 3

Simulators

3.1 Introduction

In this chapter[1], the formal definition of ROS framework is described in Section 3.2. Section 3.3 presents an example of one of the commonly used driving simulators in the ROS framework. On the other hand, the formal definition of Unity game engine is described in Section 3.4. Furthermore, Section 3.5 introduces the proposed 3D driving simulator implementation, how it the simulator utilizes both ROS and Unity softwares to achieve a better simulation environment. Finally, the chapter is summarized in few concluding remarks.

3.2 ROS Framework

ROS stands for robot operating system, which is an open source programming framework for robotics, whose development began in the late 2000s at Stanford University and Willow Garage [37]. To better understand the benefits, let's consider how robotics projects were developed before ROS existing. The kinds of commercially available systems have historically involved software control systems that were proprietary and highly specialized for their intended tasks. Each system was different, therefore lack standardization, and as a result, suffered from long development times.

ROS changed this by a framework for developing a software that facilitates and even encourages the sharing and reuse of good ideas. The goal is not to have one general purpose software programming framework that will solve all the problems with robotics development. However, ROS has maximized the utility of having open source programming framework, by addressing many of the common problems robotic developers face.

[1] Publications of the author related to the chapter are [139, 140]

ROS enables modular software development by providing a library of reusable code packages that are free and available. ROS also provides a run-time environment that supports near real-time communication between system elements and data sharing. Moreover, ROS provides programming standards, which are useful for creating and using ROS supported codes in a repeatable and reliable way. There is also a suite of ROS development tools that are helpful in monitoring, troubleshooting and visualizing the system. Last but not least, ROS has a vibrant community of tens of thousands of users and contributors all over the world, as depicted in Figure 3.1.

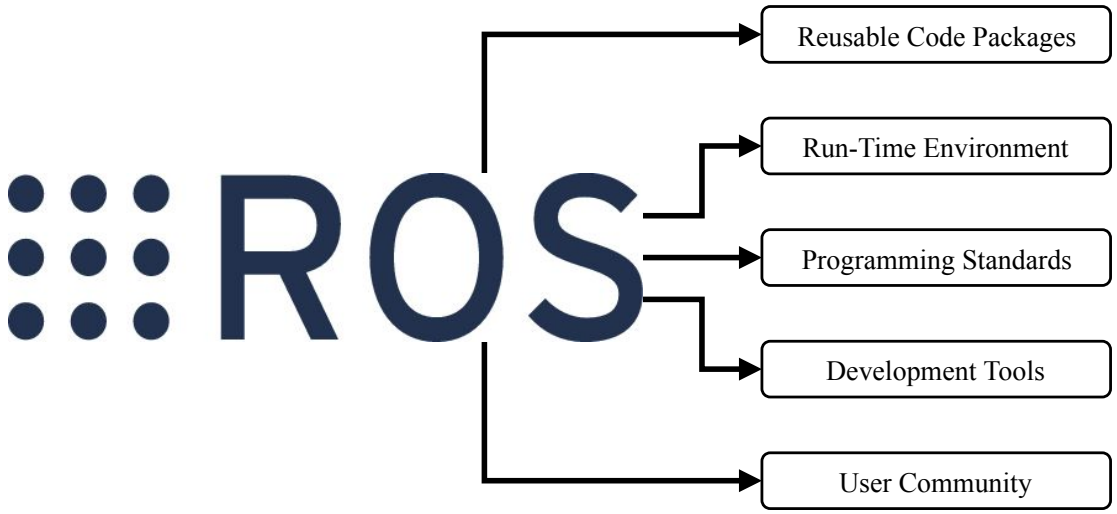


Figure 3.1 ROS framework modules

3.3 GAZEBO Simulator

In 2002, researchers from University of Southern California started the development of a new simulator environment called GAZEBO. The target was to design a high-fidelity simulator, in order to emulate robots in indoor or outdoor environments under various conditions, complementary to Stage simulator [141]. In 2013, GAZEBO was used to run the Virtual Robotics Challenge, a component in the DARPA Robotics Challenge. GAZEBO offers the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. It poses a robust physics engine, high-quality graphics, and convenient programmatic and graphical interfaces [142].

In 2017, GAZEBO creators released a simulator for a Prius in Mcity using ROS Kinetic and Gazebo 8. The simulator is called Car Demo, which allows controlling vehicle throttle,

brake, steering, and transmission through ROS topics. Additionally, all sensor data is published using ROS, and accordingly visualized with RViz [143].

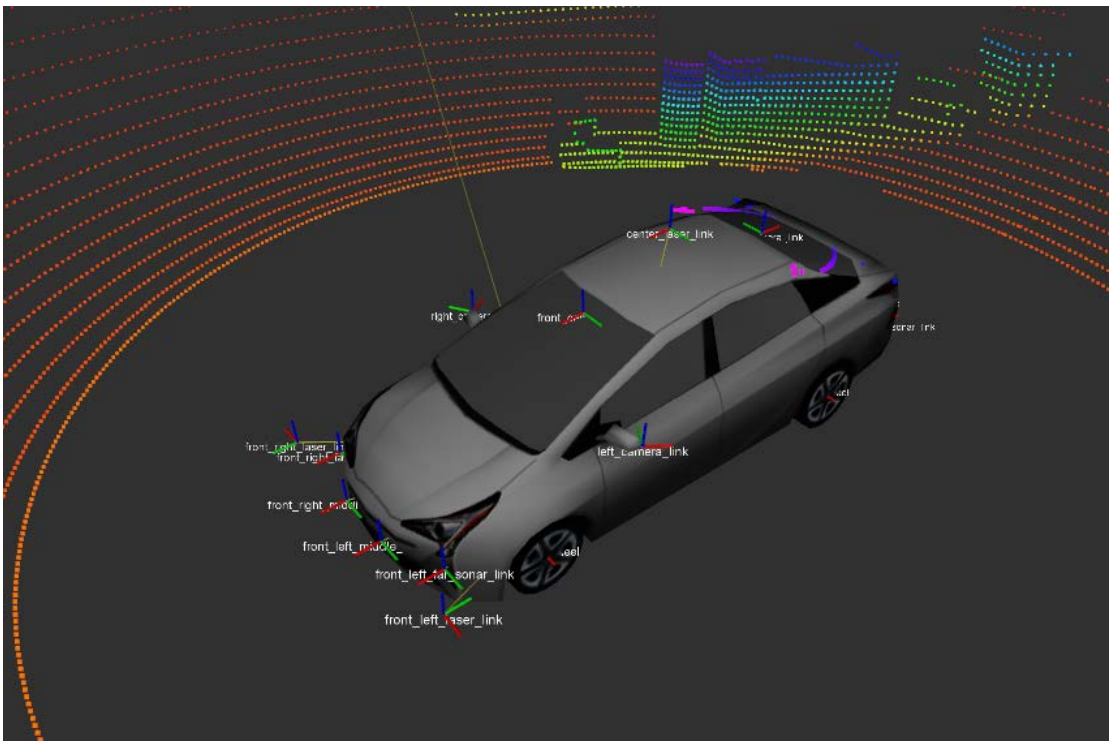


Figure 3.2 Car Demo simulator [143]

Though this simulator can be utilized to test various algorithms of localization, perception, and control among others; it is limited to the only available platform, Toyota Prius. The kinematics and dynamics of any other platform are not available, therefore the simulation does not emulate reality accurately, thus the need for a more flexible and generic simulator is justified.

3.4 Unity Game Engine

Unity Technologies released the very first version of Unity in mid-2005 targeting only OS X development [38], since then, Unity has developed and upgraded the support to more than 25 platforms including virtual and augmented reality. Unity is a platform independent game engine, where games are created by manipulating 2D or 3D objects and attaching several components to them. It has a powerful PhysX physics engine, render engine, collision detection engine among others. Furthermore, it supports high-fidelity 3D environments and allows sensors modeling [144]. Unity engine utilizes a standard mesh for the game

object, which includes the location of all vertices of the object shape. This allows the physics engine and collision detection engine to enhance the authenticity and realism of the behavior of the dynamic game objects in the simulator. Moreover, Unity render engine provides several shaders and graphics effects to enhance the authenticity and realism of the 3D virtual environment [145].

Furthermore, Unity has a wide range of online tutorials, which facilitates getting started with the development from day one. Unity has well-documented API, along with the vibrant community. Unity provides the opportunity to modify the virtual environment using an editor, or by manipulating it directly in the game window. It also provides scripting language to the developer to define behaviors and controllers.

Based on the comparison of different game engines [146], Unity is a feature-rich and highly flexible editor. As shown in Figure 3.3, it is characterized by the all-in-one editor, where it is possible to implement the project on any operating system, additionally, it supports 2D and 3D development with features and functionality needs across genres. Moreover, it has an advanced AI path-finding tool, which includes a navigation system that allows you to create NPCs that can intelligently move around the virtual environment. Another feature is the rapid iteration, where play mode is used for rapid iterative editing and the alterations in the scripts are viewed instantly. Finally, Unity has a vibrant developer community, with thousands of threads for developers to share their ideas, suggestions, information, and queries.

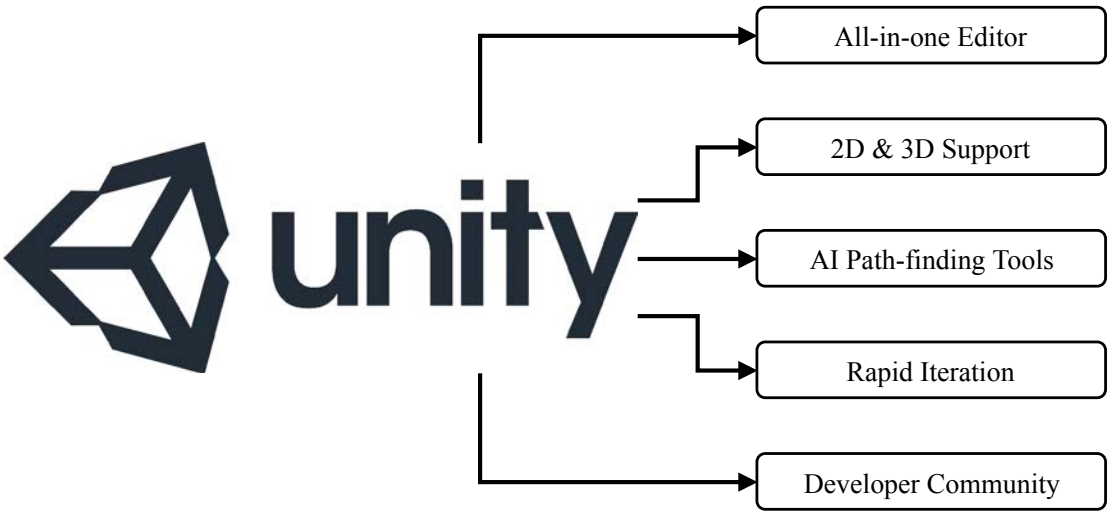


Figure 3.3 Unity game engine features

3.5 3DCoAutoSim Simulator

Based on the aforementioned information, this section presents the proposed simulator for automated vehicles, which was developed in the context of this thesis. 3DCoAutoSim is an abbreviation for "3D Simulator for Cooperative ADAS and Automated Vehicles Simulator". It is a vehicle driving simulator with high-quality 3D visualization based on Unity, which makes it possible to emulate a variety of environments. The proposed work is an extension of the capabilities presented in [147], where a driver-centric driving platform visualized the mobility behavior of other vehicles based on traffic models and a TraCI protocol allowed communication between Unity and the microscopic traffic simulator SUMO. And the work in [148], which calculated the optimal speed while approaching an intersection by retrieving the traffic light timing program from the road infrastructure, namely Traffic Light Assistance System. Finally, the work in [149], in which VANET communication capabilities were used to assess different information paradigms.

The simulator is implemented using Unity since it is a powerful 3D visualization tool, which is platform independent and has a strong physics engine. The simulator architecture hierarchy is divided into several categories; environments, scenarios, features, outputs, devices, and mode. The required configuration can be selected from the main menu, depicted in Figure 3.4.

3.5.1 Environments and Scenarios

As shown in Figure 3.4, there are several options for environments or scenarios available for selection. Selection is only for either the environments or scenarios, which will be the emulated area during the experiment.

- FHTW Environment, which is a 3D constructed map of the campus of University of Applied Sciences, Technikum Wien (FHTW) in the city of Vienna. The map construction was carried-out using the CityEngine software [150], achieving thus a one-to-one scale with high-quality visualization to emulate the real environment.
- Race City, which is provided as a testing environment by the Realistic Car Controller asset in Unity [151]. It includes several buildings, a parking lot and a large circular racing track for high-speed experiments.
- SUMO Open Street Map (OSM) based environments are dynamically configured by the user through adjusting a configuration file from anywhere in OSM and link it SUMO, further details are shown in the next sub-section.

- Scenario based selection are environments that are designed for a specific use-case, for instance, all validation results in this work were carried out using the validation scenario, which is explained in the experimental work section.

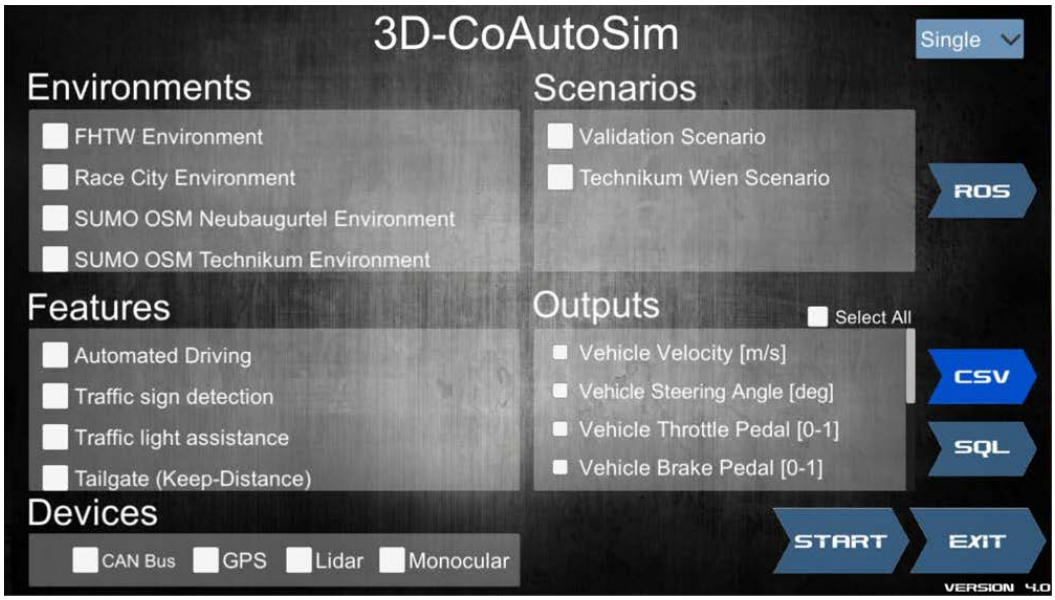


Figure 3.4 3DCoAutoSim main menu graphical interface

3.5.2 SUMO Connectivity

The communication between Unity and the microscopic traffic simulator SUMO makes it possible to use real-world road networks together with realistic traffic models to simulate a variety of driving conditions for evaluating interaction with other road users. Integrating traffic environment into the driver-centric simulator has a series of restrictions, however SUMO overcomes most of them as illustrated below.

- Macroscopic simulators lose the ability to refer to a single element in the environment, however SUMO is of a microscopic granularity nature, which allows identifying each different element.

- Time discrete and space continuous simulator. Whereas the latter can be simulated, the former is a requirement because the status of the whole simulator will be asked at a regular pace of 60Hz.

- A way to establish a bidirectional communication. Through the TraCI API, the simulator is able to provide and embody information related to each of the elements in the environment.

The SUMO OSM option requires a previously created scenario. When this option is selected and the simulation initiated, a new SUMO simulation runs in server mode. The loading operation is explained in [147]. The simulation process is as follows:

1. SUMO is started in server mode (i.e. stopped and waiting for a connection to a newly created socket in a free port), specifying the scenario the user has entered in the configuration box.
2. A new connection is made to the port from the 3DCoAutoSIM simulator.
3. All the roads are loaded to create all the network in the 3D environment.
4. All the traffic lights are loaded: the states and configuration of the traffic lights are cached in order to decrease the calls to the SUMO server.
5. All the initial cars are loaded.

The simulation is then started. For each game loop, a new simulation step is performed in the micro-simulator through the API, asking for the position of the system-controlled vehicles and communicating the current position of the user-controlled vehicle.

3.5.3 ROS Connectivity

Several libraries are required for linking the two architectures, ROS and Unity, which must be implemented in both ends. Figure 3.5 depicts the simplified general overview of the link, where the ROS framework is on the left side, with all nodes that one might have in the system, in addition to a $unity-node$, which acts as the link to subscribe and publish messages through the *rosbrdige* node from and to Unity. Whereas Unity game engine is on the right side, with all scripts one might have in the system, in addition to $ros-script$, which acts as the link to subscribe and publish messages through the *rosbridgelib* library from and to ROS. All transmitted messages between the two architectures are formatted as JSON. The detailed methodology for each architecture is described as follows.

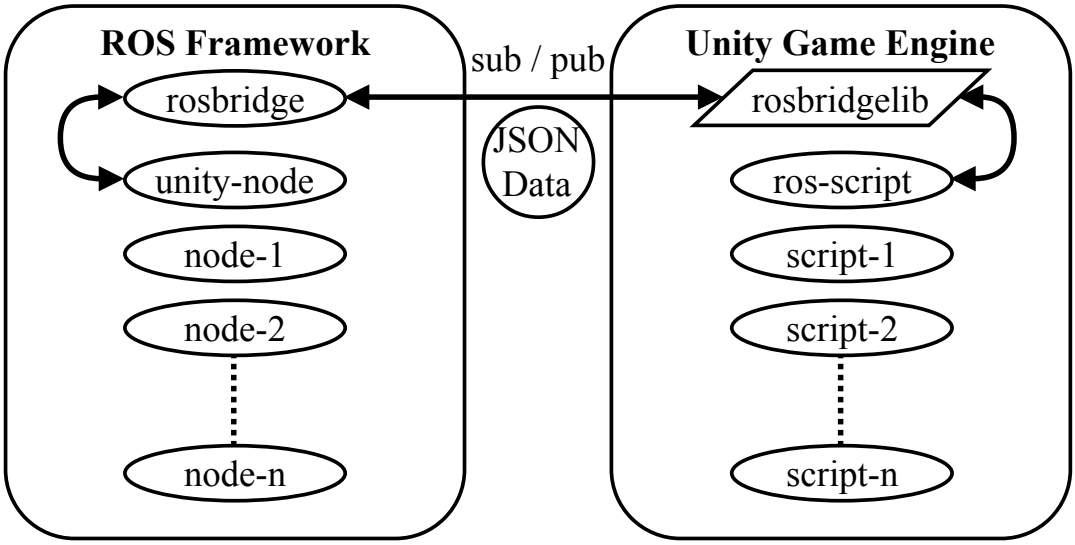


Figure 3.5 ROS and Unity link general overview

For ROS Framework

The *rosbridge* library [41] is utilized. *rosbridge* library provides a JSON API to ROS functionality for non-ROS programs. It is a Python library responsible for taking JSON strings and converting them to ROS messages, and vice versa. In this approach, WebSocket server was selected as the front end, where *rosbridge* server accepts WebSockets connections and implements the *rosbridge* protocol. Therefore, after the installation of the library, launch the $rosbridge_websocket$ node from the $rosbridge_server$ package, after adjusting the WebSocket IP and port number. Afterward, in the $unity-node$, subscribe to the messages from Unity and publish messages to Unity based on the application needs. Since the library is integrated as part of the ROS framework, it accepts any kind of standard or custom messages defined in the package dependencies.

For Unity Engine

The *rosbridgelib* library [50] is utilized. This library is to included in the Unity project assets folder, where it must be imported in the $ros-script$ file. This is the main file, which setups WebSockets connection to connect to the same IP and port number, which are adjusted by the $rosbridge_websocket$ at the ROS end. Once the connection is established, it defines the Unity publishers and subscribers. Furthermore, the $Update()$ function provides a mechanism for the callbacks on the rendering thread and deserializes JSON data into appropriate instances

of packets and messages. The library does not support every ROS message, the currently available ones are:

- *std_msgs*
- *geometry_msgs*
- *sensor_msgs*
- *nav_msgs*
- *geographic_msgs*
- *auv_msgs*

However, in the case of the necessity of new messages or custom messages, it is fairly simple to create the script files for these messages, and include them in the library folder.

3.5.4 Features

Automated Driving

Automated driving is a feature that controls the vehicle navigation to follow a set of waypoints, generated by a path planning algorithm [152]. Through the connection between ROS and Unity, the navigation commands are published from ROS based on the vehicle current location and destination point.

Traffic Sign Detection

3DCoAutoSim is also able to detect and recognize traffic signs or obstacles such as construction cones in the simulated environment, using the mounted frontal camera and the works presented in [153, 154]. The objective is to increase drivers awareness by informing them about the detected traffic signs or obstacles in the environment.

Traffic Light Assistance

Traffic light assistance option is an ADAS, which communicates with the traffic lights of the environment and provides the driver with information regarding the optimal speed to arrive at the intersection in the green phase [148].

Tailgate (Keep-Distance)

The tailgate feature is based on the works presented in [155] and [156]. The feature utilizes vehicular communication or mounted sensors, to measure the distance between the driven vehicle and the car ahead, relying on the work in [149].

3.5.5 Devices

3DCoAutoSim provides an extension API to create devices to be attached to the cars. Each new device has to be comprised at least of a prefab element, to be attached to the vehicles, and of a subclass of the Device abstract class, as described in Figure 3.6.

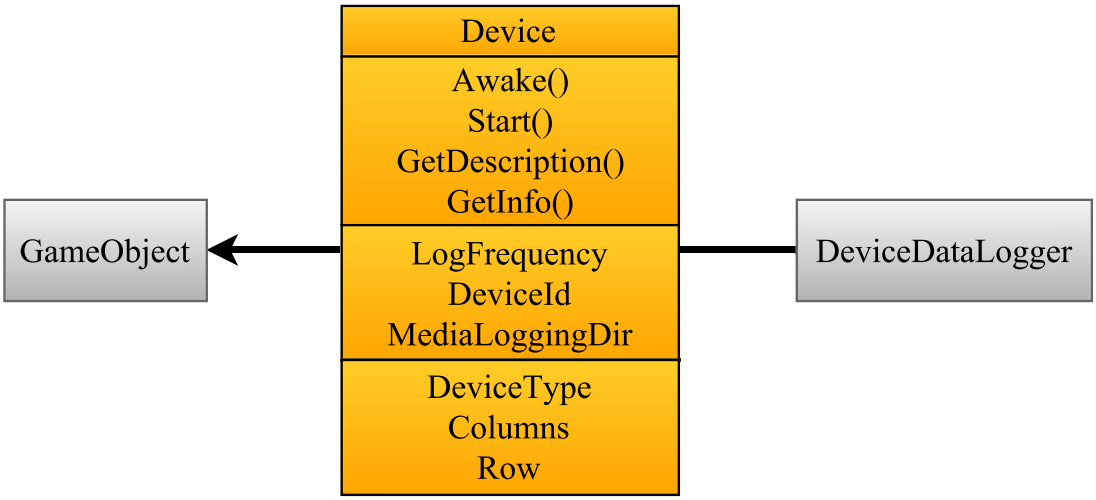


Figure 3.6 Device abstract class as the parent for all devices

The figure depicts the relationship between the devices and the associated loggers to dump their data. The reason is that the API is internally connected with the logging framework. As a consequence, any data related to a device can be easily created and logged.

3DCoAutoSim vehicles include four devices that can be enabled or disabled and allow to record their associated data for further analysis:

- **CAN Bus**. This is the only device that cannot be disabled. It gathers internal information about the vehicle, such as current speed, odometry, wheels torque and consumption. Some of these values are extracted directly from the simulator itself, whereas some others (such as the fuel consumption) are extracted from SUMO. The logging rate defaults to 10Hz.

- **GPS**. Emulates a GPS located in the middle of the vehicle. It provides the 3D position of the vehicle inside the scenario at a regular rate (defaults to 10Hz).
- **LiDAR**. Emulates a LiDAR located on top of the vehicle with the frontal position heading to the forward and with the 0-plane horizontal to the ground. It has some parameters that are configurable such as the number of planes (defaults to 8 planes), the horizontal resolution (defaults to 1deg) or the vertical separation of planes (defaults to 2.5deg). The default logging rate is 1Hz.
- **Monocular Camera**. A monocular camera which takes photos at a regular rate (defaults to 2Hz).

3.5.6 Output

3DCoAutoSim logs data for each of the devices that are used in the experiments. To this end, methods and attributes of the *Device* subclass need to be implemented, (i.e. *Row* (attribute for getting the last row of data) of *GetDescription()* (the textual information of the device object)). An overview of the logging framework that defines the methods required for the subclasses to implement is presented in Figure 3.7. In the framework *DataLogger* and *DeviceDataLogger* are the abstractions of the design for which *CsvDataLogger* and *CsvDeviceDataLogger* are concrete implementations.

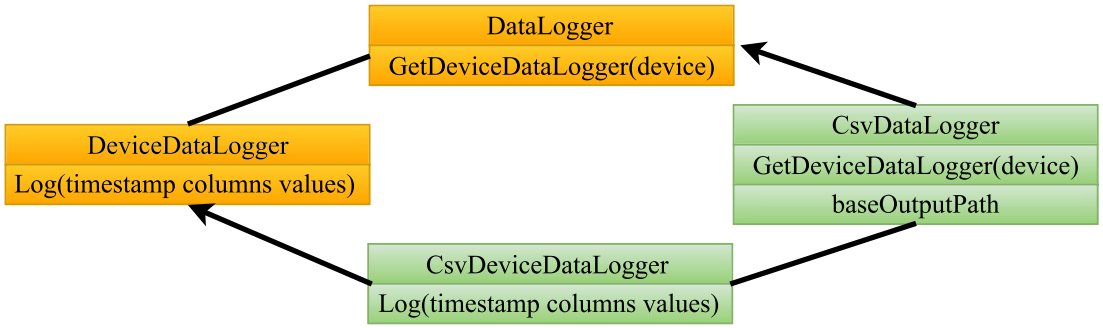


Figure 3.7 CSV logging framework design

CSV logging is a default implementation. For each experiment, it creates a new folder with the summary of the enabled devices and their configurations as well as the time stamp for starting and ending. Each of the devices creates a new CSV file with all the data logged at the specified step.

The SQL option is a further implementation. It requires a query string and the drivers' device class to connect to the database. It adds a new row to the experiment tables with the

configuration and devices. In case of missing device tables or data columns, the implementation will create them.

3.5.7 Multiple Simulators

Networked driving simulation represents an effective virtual prototyping tool, which supports the development of intelligent vehicles and accelerates system deployment [157]. Accordingly, 3DCoAutoSim supports the options of single or multiple nodes. The two systems have all aforementioned features and capabilities, in addition to Unity Multiplayer Service, which creates real-time networked instances of the simulator, each on a separate computer.

3.6 Concluding Remarks

This chapter introduced the formal definitions for the main software of this thesis, ROS framework and Unity game engine. The definition included the main features of the software and the advantages of utilizing them for the development of the work. Besides the definitions, the GAZEBO Car Demo simulator was presented. Although, it is a powerful simulator that is utilized to test various algorithms; it is limited to the only one platform. Moreover, Unity game engine has a more powerful physics engine and very high-fidelity visualization 3D environments, among others. Therefore, the concept to link ROS and Unity is essential for better emulation of reality in the simulation world. Accordingly, this chapter also introduced the developed simulator 3DCoAutoSim. The simulator has several features and capabilities, in addition to its connectivity to both SUMO and ROS. Furthermore, it allows the option of multiple user vehicles in the simulation environments, thus cooperative algorithms of automated vehicles can be analyzed, which is the main focus of this thesis.

Chapter 4

Platforms

4.1 Introduction

In this chapter[1], the three different platforms are introduced in separate sections. Section 4.2 introduces the mobile robot TurtleBot3, Section 4.3 introduces the automated drone SkyOnyx, and Section 4.4 introduces the automated vehicles iCab. The first two platforms are briefly described, highlighting their main features and capabilities, since their purpose was to validate the heterogeneity of proposed work. However, the iCab is the main platform of this thesis, accordingly, the detailed transformation of the platform from regular to the automated vehicle is described, followed by a comprehensive description for the designed software architecture and all implemented systems for the automation objective. Finally, the chapter is summarized in few concluding remarks.

4.2 TurtleBot3

In this section, hardware and software descriptions of TurtleBot3 platform are presented.

4.2.1 Hardware Description

TurtleBot is a series of inexpensive robots developed specifically for ROS. The first TurtleBot prototype was created by Willow Garage in 2010. Willow Garage is a group that develops hardware and open source software for robotic applications. TurtleBot name comes from the ROS tutorial simulator, which was named after the cursor name of the Logo software. TurtleBot2 was released in 2012 and TurtleBot3 in 2017 [48]. TurtleBot3 designs are clearly

[1] Publications of the author related to the chapter are [158–166, 152, 167–171]

smaller than the previous TurtleBots. TurtleBots included an embedded computer, which was replaced by TurtleBot3 single PCB computer, Raspberry Pi 3 [172]. The robot has also been made very modular, as the whole frame has built-in screws for easier mounting [173]. TurtleBot3 is available as two models; "Burger" is smaller, and "Waffle" is larger. The models chassis consists of identical mounting plates, but have different components. The most technical difference between the models is the 3D camera included in "Waffle" TurtleBot3 [174]. Both platforms are shown in Figure 4.1.

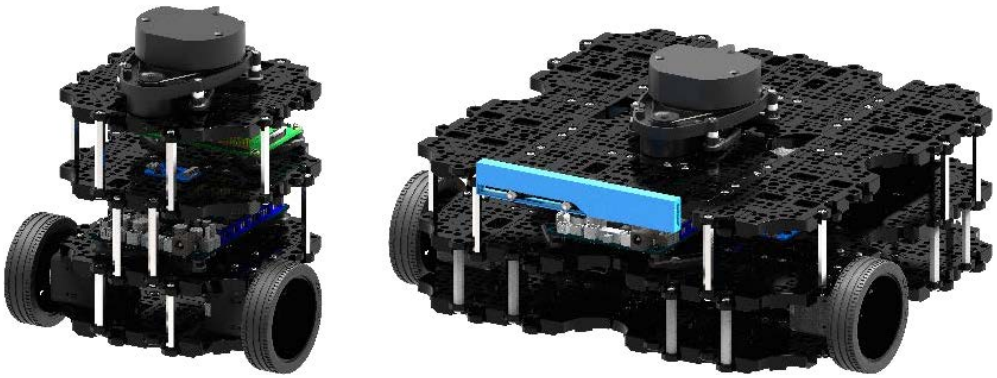


Figure 4.1 TurtleBot3 platforms; "Burger" (left), and "Waffle" (right) [174]

TurtleBot3 uses XL430-W250-T DYNAMIXEL motors for the differential motion, which are embedded with optical encoders. The motors are controlled by the STM32F7 OpenCR board. Furthermore, it is equipped with a 360°Laser Distance Sensor LDS-01(HLS-LFCD2). The sensor is capable of measuring unobtrusive and opaque objects at a minimum of about 12cm and a maximum of about 3.5m away. The sensor measures only two-dimensional, meaning it only detects objects in the same plane.

4.2.2 Software Description

There are several ROS packages included in TurtleBot3 architecture, as depicted in Figure 4.2. The *gmapping* package provides laser-based SLAM, it creates a 2D occupancy grid map from laser and pose data collected by the robot [175]. Later the *navigation* stack package takes in information from odometry, sensor streams, and a goal pose and outputs safe velocity commands that are sent to a mobile base [176]. TurtleBot3 utilizes the *amcl* package for probabilistic localization, while the *move_base* package is used for the attempt to reach a given goal pose in the world. Last but not least, the *dwa_local_planner* package provides an implementation of the Dynamic Window Approach for local robot navigation on a plane. A detailed description of the software architecture is provided in the eManual [177].

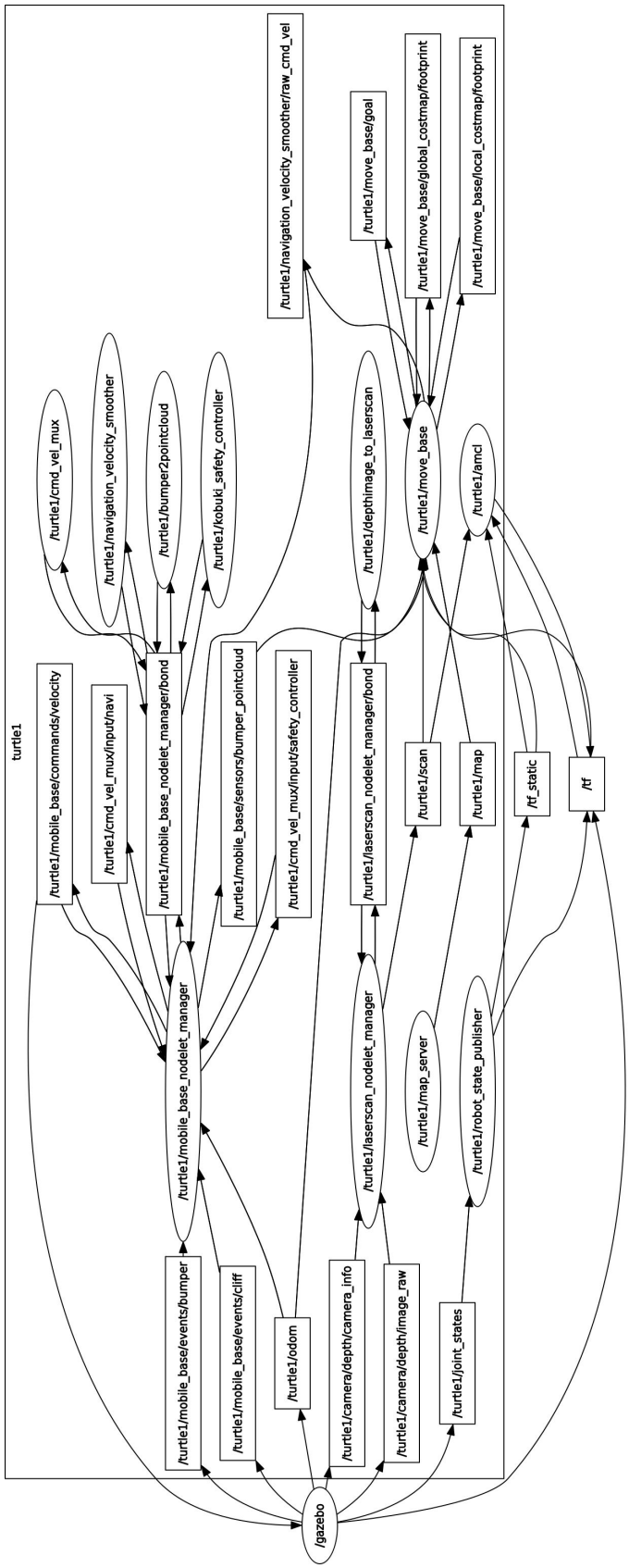
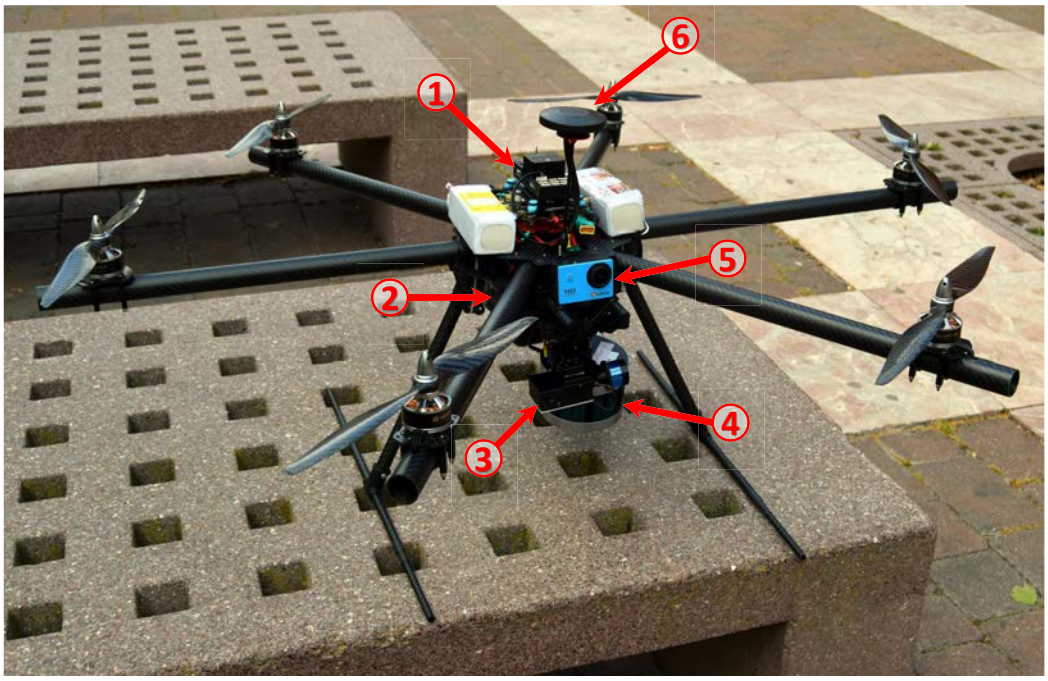


Figure 4.2 TurtleBot3 architecture



1- Flight Controller
2- On-board Computer
3- Downward Camera
4- Lidar
5- Frontal Camera
6- GPS

Figure 4.3 SkyOnyx platform

4.3.1 Hardware Description

As shown in Figure 4.3, SkyOnyx body is a carbon fiber hexacopter of total weight 4.5kg, based on Pixhawk 2 autopilot. SkyOnyx is equipped with three main navigation sensors; GPS, IMU, and barometer. Additionally, it has two on-board SJCAM SJ4000 monocular cameras, which provide 640×480 RGB images. All the processing is performed on-board by an embedded computer; Intel NUC computer, which has an Intel i7-7567U CPU at 3.5GHz CPU and 8GB RAM. The software has been developed and integrated with ROS, under Ubuntu 16.04 LTS operating system. A full description of the SkyOnyx hardware is detailed in Table 4.1.

Table 4.1 SkyOnyx hardware description

Hardware	Description
Structure	A hexacopter with carbon fiber body. It has T-Motor KV 470 motors and F 30A Dshot ESCs, with total endurance of 15 min and total weight of 4.5kg
Flight Controller	**Pixhawk 2 Cube:** **Processor** 32-bit ARM Cortex M4 core with FPU 168 Mhz/256 KB RAM/2 MB Flash 32-bit failsafe co-processor **Sensors** Three redundant IMU sensors InvenSense MPU9250 and ICM20948 as first and third IMUST Micro L3GD20+LSM303D as backup IMU Two redundant MS5611 barometers
On-board Computer	Intel NUC NUC7i7BNH: Intel Dual-Core i7-7567U @ 3.5GHz with Turbo Boost upto 4.0GHz, 4MB Cache, 8GB DDR4 2133MHz WD Green M.2, 120 GB SSD - SATA 6 Gb/s Intel Iris Plus Graphics 650
Camera	Two monocular cameras, one downward and one forward; both are SJCAM SJ4000 with 640 x 480 RGB images
Lidar	Velodyne VLP-16: Dual Returns 16 Channels, 100m Range 360°horizontal and 15°Vertical FOV
GPS	Concurrent reception of up to 3 GNSS (GPS, Galileo, GLONASS, BeiDou) Industry leading 167 dBm navigation sensitivity HMC5983 MAG, and LIS3MDL Mag

4.3.2 Software Description

The presented architecture of the SkyOnyx platform exchanges the information between the subsystems based on the task. Figure 4.4 shows the overview of the whole system.

The data acquisition phase is the responsible of obtaining the raw data from the mounted sensors; such as monocular cameras, IMU, GPS, among others, and transfers it to the different modules in the system.

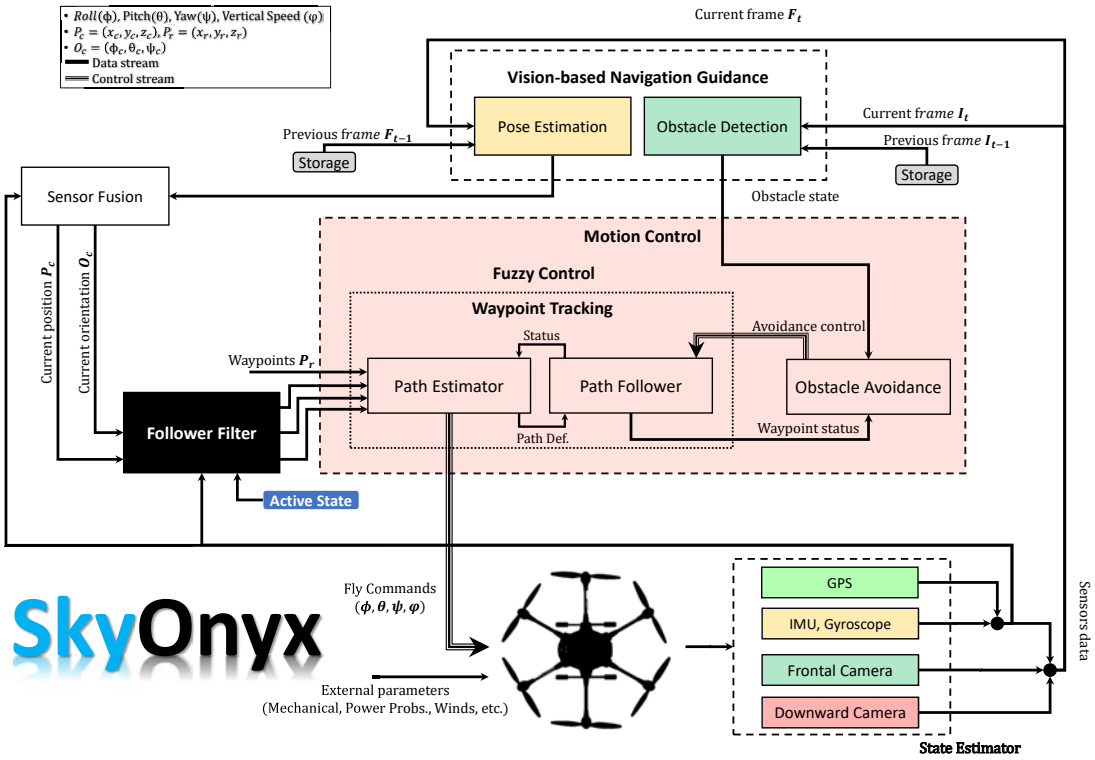


Figure 4.4 SkyOnyx architecture [178]

Localization

The UAV flies in indoor and outdoor environments with the ability to localize itself accurately. In the proposed platform, the localization system is based on the fusion of three main sensors (downward monocular camera, IMU, and GPS), with the ability to fuse other sensors data as well.

Visual Odometry The pose estimation solution for estimating 6-DOF attitude of the UAV is presented in [179]. This is achieved by solving the different homographies *world-to-frame* and *frame-to-frame*; in order to estimate the relation between the detected points in the image and its positions in the 3D world relative to the UAV position.

IMU Odometry One of the essential sensors for the UAV localization is the IMU, which provides sensor readings for acceleration, gyroscopic and magnetic data, granting SkyOnyx with 9-DOFs of sensorial awareness. The advantage of this type of sensor is the fast sampling rate and it provides a lot of information about the motion of the vehicle although its precision is not as accurate as desired.

GPS Odometry GPS is considered as one of the most known approaches that are used for outdoor localization [180]. It determines the absolute position and orientation in the

3D space according to the respective positioning of at least three satellites, from a total of 24, orbiting around the Earth. The GPS has integrated a compass to correct the orientation measured by the GPS signal [181].

Odometries Fusion Due to the inaccuracy or the noise in the information of the sensors; which leads to worsening the accuracy of the localization; fusing the obtained information from multiple sensors to improve the accuracy of the localization system is very common. Accordingly, an Extended H_∞ filter is used to fuse multiple odometries (GPS, IMU, and visual) for localizing the aerial vehicle SkyOnyx. The fusion algorithm is extended work of [168].

Perception

Another challenging problem in the domain of autonomous aerial vehicles is the designing of a robust real-time obstacle detection and avoidance system. The perception module implemented in SkyOnyx is mainly dealing with the obstacle detection problem, providing a bio-inspired method that mimics the human behavior of analyzing the size states of the approaching obstacles using monocular camera [182].

Planning

Furthermore, the planning module presents an approach to perform automated flights from the start point to the destination. This module is divided into two stages; a planning stage with a robust guidance algorithm, that generates the waypoints to determine the flight path based on environment, and the situated obstacles. And a waypoint tracking stage with a strategy of following and swapping the generated waypoints efficiently [158].

Control

Furthermore, the ability to fly safely from the start point to the destination is crucial in the domain of autonomous UAVs. Besides a robust guidance and path planning algorithms, a stable control system to perform the path following as well as the obstacle avoidance maneuver is required. The control module is consists of two systems; one is responsible fro the flight stability and waypoint tracking maneuvers, while the second control is responsible for obstacle avoidance maneuvers. The Fuzzy Logic Control (FLC) is formulated to control the stability of the UAV in the *Hover* flight mode, as well as to track and follow the predefined waypoints. Two parallel controllers are implemented; the first one is to control the heading angle and the altitude of the UAV, whilst the second one is implemented to control the angles (pitch, roll), and the vertical velocities in (x, y)-axes [183].

4.4 iCab

In this section, hardware and software descriptions of iCab platforms are presented.

4.4.1 Hardware Description

iCab project consists of two electric golf-carts, which are shown in Figure 4.5. iCab is an abbreviation for "Intelligent Campus Automobile", which utilizes multiple automated UGV for shared mobility application. The vehicles are modified electrically, electronically and mechanically for the purpose of self-driving to carry-out users transportation requests from one point to another, with minimal human intervention. The vehicles modifications and hardware description are presented below, for both platforms.

Figure 4.5 Proposed platforms, red iCab-2 is an EZ-GO RXV 2008 (left) and blue iCab-1 is an EZ-GO RXV 2009 (right)

Steering System

In order to design the vehicle steer by wire system, the steering wheel was replaced with an actuator and an absolute encoder systems, as shown in Figure 4.6. The steering actuator is a *Parvalux PM60 LWS* brushed DC motor, which has the specifications detailed in Table 4.2. It has a spur gear ratio of r_1 of $6:1$, a worm gear ration of r_2 of $\frac{25}{3}:1$ and an estimated motor efficiency η of 80%. The motor had mounted to it a TEKEL TKM60 absolute encoder, which had a maximum resolution of 8192 positions per 4096 turns.

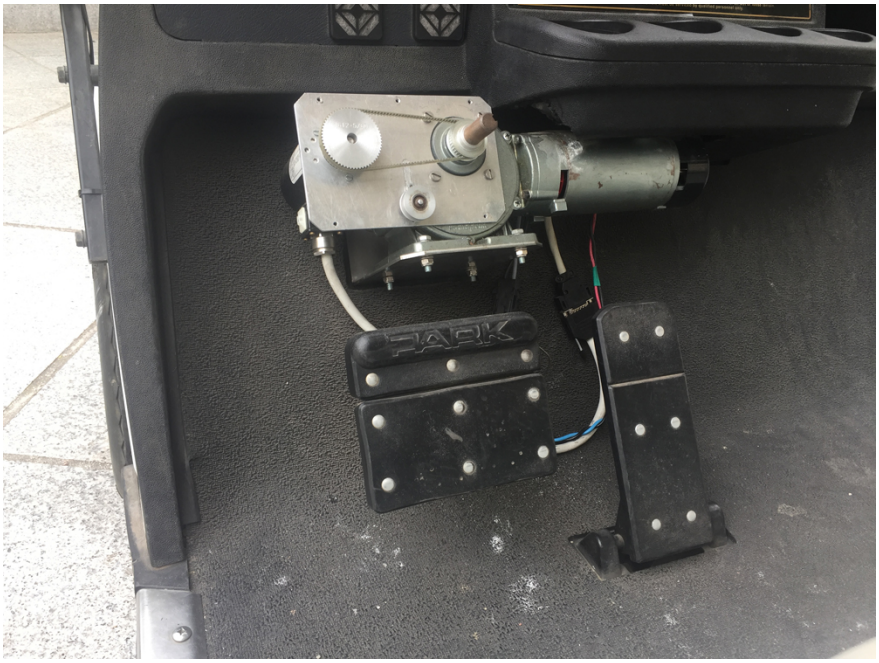


Figure 4.6 iCab steering actuator

Table 4.2 Steering actuator specifications

Parameter	Value	Unit
K_t	0.054	$\mathrm{N\,m/A}$
R_m	1.25	Ω
L_m	10^{-6}	H

As shown in Figure 4.7, the dynamic model of the vehicle is the based on Ackermann steering principle [184]. The experimental relation between the column steering angle δ_c and the reduced bicycle model steering angle δ is estimated in Equation 4.1.

$$\delta = 0.056 \times \delta_c \tag{4.1}$$

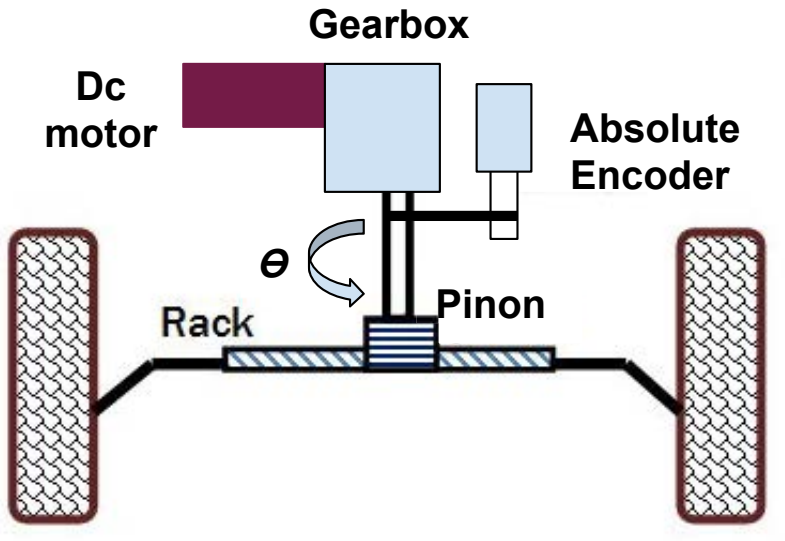


Figure 4.7 Ackermann steering schematic [185]

The system was modeled using Simscape, the MATLAB, and Simulink physical modeling tool. The simulated response shows that deceleration is high and that the system reaches zero velocity in no time. Accordingly, a rotational hard stop block was added to the model to account for the backlash between the worm and worm-gear, which is superior among all reductions. The angle of backlash was found to be 0.227°. Accordingly, the controller was chosen to minimize overshoot, with the tuned gains are: K_P is 65, K_I is 0, and K_D is 10.

Drive System

In order to design the vehicle drive by wire system, the throttle pedal was deactivated and the control is applied directly to the traction brushed DC motor. iCab platforms traction motors the 36 VDC, Series Wound, Non Vented 2.5 hp (1.9 kW) @ 2700 RPM Brazed Armature and Solid Copper Windings. The motor is composed of a rotor, represented and a stator. Moreover, a Heidenhain ROD426 rotary encoder is mounted on the motor main shaft, it has up-to 10,000 incremental signals lines.

In this thesis, the drive by wire system is divided into two levels. The first level is the low-level controller, by utilizing a microcontroller to replace the accelerator box. A fine-tuned PID controller was designed and implemented in the microcontroller to control the values of the stator and rotor, the work is presented in [186].

The second level is velocity controller, and in order to determine the gains of the controller, a model of the system is designed in MATLAB and Simulink. The input of the controller

corresponds to the reference velocity from the navigation controller, and the feedback signal corresponds to measured velocity by the encoder, and finally, the output corresponds to the Pulse Width Modulation (PWM) value that controls the rotor speed.

In order to have a complete drive by wire system, the velocity controller must be linked with the brake by wire system, which is explained in detail in the next subsection.

Brake System

In order to design the brake by wire system, the brake pedal action must be activated by an external electronic actuation, however maintaining the brake pedal functionality, as an external safety option for on-board passengers. iCab platforms have a set of self-adjusting drum brakes on the rear axle type, which means that the brakes adjust themselves depending on the amount of wear that occurs in the friction element. Thus the friction element always maintains a constant gap between itself and the hub. In order to externally activates the braking power, a linear actuator is attached to the vehicle frame by a metal casing. Therefore, when the brake signal is sent, the linear actuator pushes the equalizer, which pulls the brake cable and stops the vehicle. The actuator is the MCCA100-12V-50mm linear motor, which has a speed of 2mm$/$s and 1000N maximum load. Attached to the actuator, a linear position sensor, the Novotechnik TR100, which has an accuracy of 0.002mm, which is shown in Figure 4.8.

Figure 4.8 iCab braking actuator

Furthermore, the braking deceleration that a vehicle can achieve is defined as the product of application level and the brake gains of torque. This deceleration brings up a consequence related to the vehicle load which is translated into a weight transfer. It basically modifies

the position of the center of mass of the vehicle, delivering the majority of the weight to the front axle of it, which is represented in Figure 4.9.

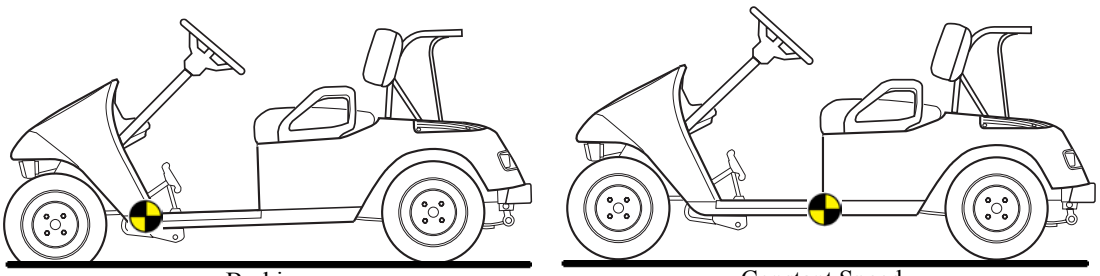


Figure 4.9 Vehicle center of mass when braking (left) or at constant speed (right)

Taking the aforementioned modeling for both drive and brake by wire systems into consideration, the braking force is estimated using Equation 4.2. Therefore, the required braking power is based on the change in vehicle velocity, the change in the vehicle weight based on the number of passengers aboard, and type of road.

$$F_b = W_t * \left(\frac{a}{g} - f_r \right) \tag{4.2}$$

where F_b is the vehicle braking force, W_t is the vehicle total weight, a is the vehicle acceleration, g is the gravity of Earth, and f_r is the vehicle rolling friction coefficient.

The brakes are now capable of reducing speed accordingly to the needs of the real environment. When designing the controller, the braking torque applied to the vehicle was a parameter, which could be adjusted to reduce velocity at a faster or slower rate, based on the detected difference between speed values. It has been created in order to provide and ensure commodity and safety for the user, which will not have any control over the vehicle except for an emergency button, which deactivates all the mechanisms of the vehicle.

At this stage, the iCab platform has all the necessary actuators installed for executing the driving behavior via controllers. Thus, the by wire systems represent the electrical, electronical and mechanical modification to the platform, which were carried out for both golf-carts. The verification of the proposed controllers and systems is presented in [187–189]. Furthermore, the schematics for the electrical wirings, electronics configuration and acquisition driver packages is presented in [190].

On-board Devices

The two platforms are almost identical in terms of the on-board devices. Figure 4.10 shows the side-view of iCab-1 platform, indicating the location of all on-board devices. The description of all the devices is summarized in Table 4.3.

Table 4.3 iCab platform on-board devices description

Category	Hardware	Description	
		iCab-1	iCab-2
Computers	Embedded computer 1	AAEON AEC-6877[2]	
	Embedded computer 2	Intel NUC6i7KYK[3]	
Sensors	Laser rangefinder	SICK LMS291[4]	
	Stereo camera	Bumblebee 2 1394a[5]	Bumblebee XB3 1394b[6]
	Lidar	Velodyne VLP-16 Puck[7]	
	GPS and compass	3DR UBlox Module[8]	
	Kinect	Kinect Sensor[9]	
	Ultrasonic sensors	MaxBotix MB8450[10]	
Auxiliaries	Primary touchscreen	Hannspree HT 231 HPB 23” Touchscreen[11]	
	Secondary touchscreen	Xenarc 705TSV 7" Touchscreen[12]	
	Router	TP-LINK Archer AC750 4G LTE[13]	
	Light	Rotating Warning Light	
	Buzzer	Warning Buzzer	
	Speaker	Stereo Speaker	

[2]http://www.aaeon.com/en/p/fanless-embedded-computers-aec-6877
[3]https://www.intel.com/content/www/us/en/nuc/nuc-kit-nuc6i7kyk-features-configurations.html
[4]https://www.sick.com/ag/en/detection-and-ranging-solutions/2d-lidar-sensors/lms2xx/c/g91905
[5]https://www.ptgrey.com/bumblebee2-firewire-stereo-vision-camera-systems
[6]https://www.ptgrey.com/bumblebee-xb3-1394b-stereo-vision-camera-systems-2
[7]http://velodynelidar.com/vlp-16.html
[8]http://ardupilot.org/copter/docs/common-installing-3dr-ublox-gps-compass-module.html
[9]https://msdn.microsoft.com/en-us/library/hh438998.aspx?f=255&MSPPError=-2147217396
[10]https://www.digikey.com/catalog/en/partgroup/mb8450-usb-carsonar-wr/72338
[11]http://www.hannspree.eu/en/monitors-range/ht-series/ht231hpb
[12]https://www.xenarc.com/705TSV.html
[13]https://www.tp-link.com/uk/products/details/cat-9_Archer-C2.html

Figure 4.10 iCab-1 platform on-board devices side and front views

Every single device is used for the software architecture to achieve the vehicle automation. The computers are running Ubuntu 16.04 LTS operating system and hold the designed ROS-based architecture, which is explained below. All sensors are used for localization, mapping, and environment perception, furthermore, the data is fused for a better overall performance of the system. Finally, the auxiliaries are used for the platform passenger or for the environment users, as information and warning devices. All the aforementioned modifications to the platforms were developed in the framework of this thesis, along with the mounting and wiring for the devices.

4.4.2 Software Description

Figure 4.11 presents the overall software architecture for the iCab platform. The architecture is designed in a ROS framework for the automation of the platform. It consists of five main layers; acquisition, processing, decision, control, and actuation. Each layer consists of several packages to send and receive data and connects the whole architecture as a single entity.

Figure 4.11 iCab overall software architecture

First, the acquisition layer, where the drivers packages provide a software interface to the on-board sensors, enabling the operating system to access the device functions and publish the data in the ROS messages format to the second layer, the processing layer. The processing layer holds the core systems for vehicle automation, where multiple localization systems are implemented for the estimation of the vehicle position and orientation in the environment, combined with mapping systems for the environment understanding as both globally and locally. Additionally, the perception systems are implemented in the processing layer for the obstacles detection and classification. Moreover, several communication schemes are implemented, which are utilized in the cooperation and coordination systems. Upon finalizing the processing of all data from the sensors, the decisions are made in the third layer, where the global and local planning systems are implemented. Afterward, the control layer takes the planning output and estimate the necessary actions for the vehicle movement, in both lateral and longitudinal. Finally, the output of the control layer is connected to the actuation layer, which has the steer by wire, drive by wire, and brake by wire systems for the low-level control. The description for all implemented packages is presented in the below subsections.

Auxiliaries

There are two auxiliaries packages in the iCab platform, the Graphical User Interface (GUI) package and the sound manager package, which are explained below. The other auxiliary devices, such as the warning buzzer and light are activated using dashboard switches during the operation of the vehicle. These devices ensure additional safety measures for the road users surrounding the vehicle.

Graphical User Interface: This package displays the status and states of the platform, displays informative messages to the passengers related to the journey, and takes the passengers input for their destination or to override the automation process and go back to manual control. The GUI is developed in C++ using Qt graphics libraries[14]. The GUI consists of five tabs, the "Main" tab, which is shown in Figure 4.12, it presents the necessary operative information. The "SLAM" tab presents the fused localization and mapping outputs, the "Odometries" tab presents all the different localization systems outputs, the "Obstacles" tab presents the obstacle detection and classification systems outputs, and finally the "ROS" tab includes the parameters for the link of the GUI with the framework. The detailed description of the implementation of the GUI and its packages are presented in [190].

[14]https://doc.qt.io/qt-5.10/topics-graphics.html

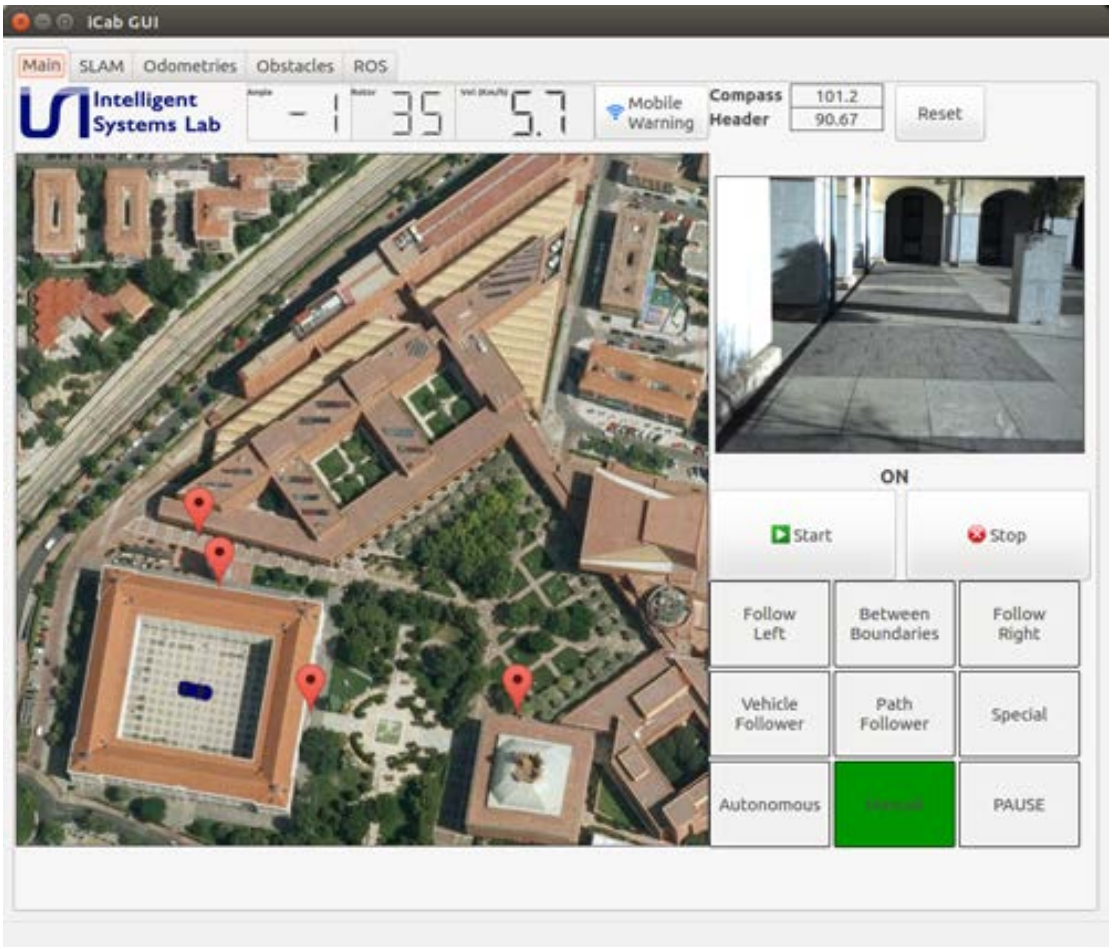


Figure 4.12 iCab GUI

Sound Manager: The sound manager package is responsible for generating audio messages to the passengers. The messages are either informative about the environment or for warning purposes. The informative messages include the affirmation to the inputs commands from the passenger and stating the vehicle current driving mode. On the other hand, the warning messages include the information about the detected obstacles. Though the simplicity of this package, it is essential for a complete platform user interactions. The package is developed in C++ and connected to all the required topics to trigger the audio messages in the ROS framework.

Communication

The communication packages are implemented based on developed communication schemes explained in Chapter 5.

Multi-master: This package is necessary for the inter-vehicular communication, since it ensures that each platform has its own ROS, thus guaranteeing the system distributed paradigm. The *multimaster_fkie* package is based on the work in [191], where authors combined the essential packages to create and control a multi-master network in ROS framework. The proposed design requires the vehicles to be under the same network, which is achieved through the development of Virtual Private Network (VPN) and detailed in Chapter 5. The objective of this approach is to use VPN over the available internet connection. Therefore, a secure multi-cast connection is initiated and the V2X communication schemes are created and summarized in Chapter 5.

Pedestrian Warning Manager: This package is necessary for the pedestrian communication, where its main purpose is to communicate with the developed smartphone application for collision prediction and warning of Vulnerable Road Users (VRU). The application detailed description is presented in Chapter 5. The package receives the estimated collision coordinates from the smartphone and uses it to improve the environment perception by detecting a pedestrian who is not in the line of sight.

Webclient Manager: This package is necessary for the infrastructure communication, where its main purpose is to communicate with the developed webserver. The webserver detailed description is presented in Chapter 5. The webclient manager package gets the list of requests from the webserver through JSON messages and decodes to ROS messages to be published to the framework. Moreover, it sends the vehicle stats to the webserver through JSON messages after encoding the subscribed ROS messages. This way it ensures bidirectional communication with the webserver and maintaining the ROS standards for the inter-packages communication.

Control

The vehicle control architecture consists of three tiers based on the work presented in [192]. The architecture of the control tiers is presented in Figure 4.13, starting by the reactive tier and its connection to the environment, after taking the tasks from the sequencer tier, which takes the orders from the high deliberative tier. This control architecture is flexible and allows the addition of more features and or the modification of the interlaced algorithms to improve the output results. The detailed description of the control architecture and its packages are presented in [190], and brief descriptions of the tiers are as follows:

- **Reactive Tier:** it is the lowest control layer, in which the vehicle lateral and longitudinal controllers are implemented for the movement of the automated vehicle. The inputs are obtained from middle layer, the sequencer tier. These inputs are transferred one

after the other to estimate the necessary movement outputs for the steer by wire system, drive by wire system, and brake by wire system for the actuation process [190].

- **Sequencer Tier:** it is the connection layer between the high-level and low-level controllers. The inputs are obtained from the deliberative tier one after the other as a complex task, and the outputs are sent to the reactive tier. The objective is to decompose the high-level tasks to reactive ones. Furthermore, an obstacle threat interrupter is simultaneously executed for the detection of risky situations in the environment [190].
- **Deliberative Tier:** it is the top layer to coordinate the required actions for the overall system. After data processing stage, the layer outputs are sent one after the other to the sequencer tier for decomposition. This layer is responsible for the environmental awareness before taking the high-level decisions [190].

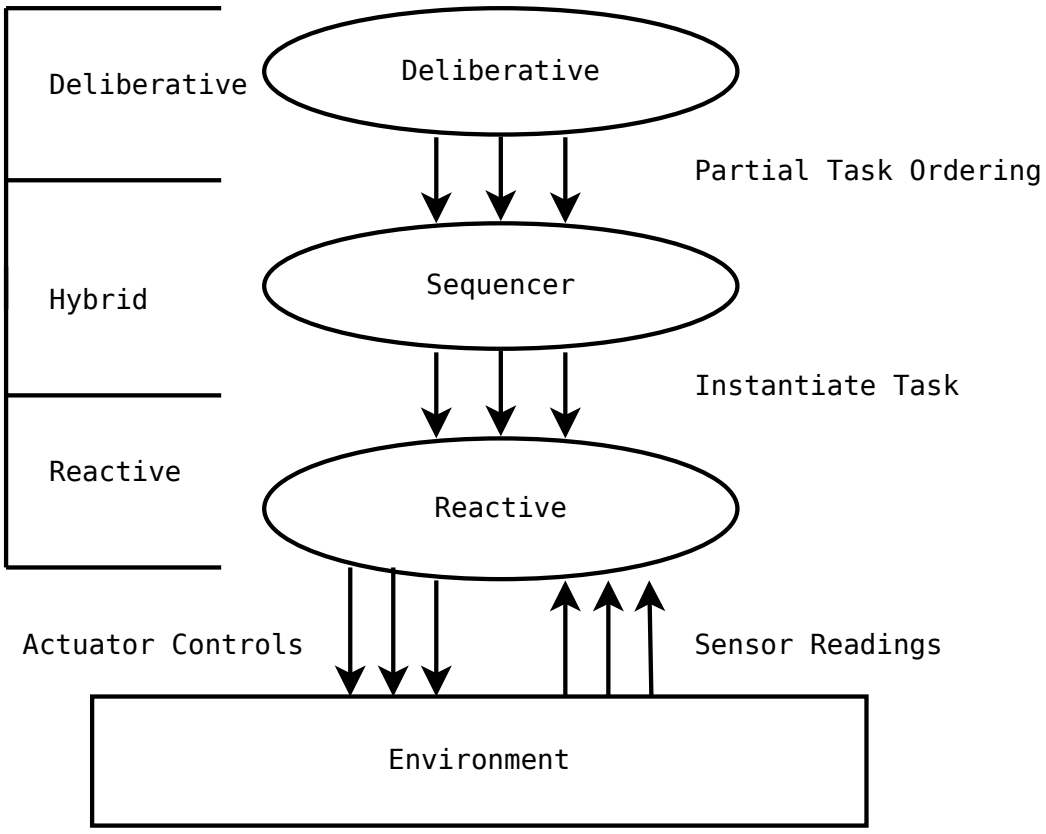


Figure 4.13 iCab control three-tiers architecture

Cooperation

Since this thesis presents multiple automated vehicles, the cooperation packages are essential to execute the necessary algorithms for coordination among the vehicles.

Platooning: This package is responsible for the implementation of the platooning algorithm presented in Section 5.3. The package subscribes to the states and statuses of the vehicles in the system in order to estimate the necessary commands maintaining the platoon in order.

MRTA: This package is responsible for the implementation of the task allocation algorithm presented in Chapter 6. The package subscribes to the states of the vehicles in the system in order to execute the allocation algorithm and assign the tasks to each vehicle accordingly.

Localization

The localization system is considered as the most crucial steps in the vehicle automation, therefore redundancy of several approaches is required and after the fusion of all these approaches to achieve the most accurate results.

GPS and Compass Odometry: This package subscribes to the data from the on-board GPS and compass module and estimates the global position and orientation accordingly. This is the only package that estimates global localization since the algorithm does not measure the vehicle displacements with respect to a fixed reference point, however, it provides the absolute position on the terrestrial surface of the receiving device. The developed algorithm is based on the work presented in [193], which filters and estimates the raw data to obtain longitude and latitude coordinates. However, the accuracy of the results is not perfect, especially in urban environments. Accordingly, the package converts the longitude and latitude coordinates to local coordinates, and the compass reading to a local heading angle. Therefore, the output odometry can be integrated into the fusion algorithm.

Initial Pose Estimation: The kidnapped robot problem commonly refers to a situation where an automated robot in operation is starting in an arbitrary location. This problem is frequent in many applications and therefore and several approaches were proposed for solving it [194–196]. However these solutions require expensive equipment to be installed in the environment, on the other hand, widespread solution based on particle filters is considered as a proper alternative [197]. The main advantage of this approach is the usage of the on-board sensors of the vehicle, such as a lidar sensor or a stereo camera. Therefore, in the proposed approach the lidar pointcloud is processed using PointCloudLibrary and LibPointMatcher. These libraries are used for preprocessing of the pointcloud before proceeding to the search

of the initial pose estimation. A detailed description of the developed algorithm is presented in [198]. The output from this package is the global offset for the vehicle to integrate with all the following local localization systems.

Visual Odometry: This package utilizes the stereo images and computes an estimated ego location of the vehicle based on the work presented in [199]. This method is based on tracking the road feature points frame by frame, in order to estimate the movement of the vehicle, avoiding outliers from dynamic obstacles. The road profile is used to obtain the world coordinates of the feature points as a unique function of its left image coordinates. Though the algorithm obtains good results in most of the scenarios, it is not fully reliable, since the lighting condition of the environment affects it drastically.

IMU Odometry: The IMU is used to determine the accelerations as well as the orientation of the vehicle in the 3D space. Accordingly, using this information, the pose vector $\hat{\mathbf{x}} \in \mathscr{R}^6$ of the vehicle, which consists of the three spatial coordinates as well as the Euler angles can be determined according to the kinematic model in Equation 4.3.

$$\begin{aligned} \hat{\mathbf{x}}_\mathbf{k} &= \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \dot{\hat{\mathbf{x}}}_k, \mathbf{u}_k) \\ &= \mathbf{x}_{k-1} + \dot{\mathbf{x}}_k dt + \ddot{\mathbf{x}}_k \frac{dt^2}{2} \end{aligned} \tag{4.3}$$

where dt is the sampling time and $\dot{\hat{\mathbf{x}}}_{k-1}, \mathbf{u}_k$ are shown in Equations (4.4) and (4.5)

$$\dot{\hat{\mathbf{x}}}_k = \mathbf{f}(\dot{\hat{\mathbf{x}}}_{k-1}, \mathbf{u}_k) = \dot{\mathbf{x}}_{k-1} + \ddot{\mathbf{x}}_k dt \tag{4.4}$$

$$\mathbf{u}_k = \begin{bmatrix} \ddot{x}_k \\ \ddot{y}_k \\ \ddot{z}_k \\ \dot{\alpha}_k \\ \dot{\beta}_k \\ \dot{\gamma}_k \end{bmatrix} \tag{4.5}$$

where α, β and γ are the roll, pitch and yaw angles respectively.

Therefore, the package follows the presented model and estimates the vehicle position and orientation. However, the IMU odometry suffers from a very high error and are considered to have completely diverged.

Encoder Odometry: The pose of the vehicle is computed recursively, using the previous pose $\mathbf{x}_{k-1}$ and the input to the vehicle $\mathbf{u}_k$, as illustrated in Equations (4.6), (4.7) and (4.8).

$$\begin{aligned}\hat{\mathbf{x}}_\mathbf{k} &= \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k) \\ &= \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} \Delta d_k^e cos(\theta_{k-1} + \Delta\theta_k^e/2) \\ \Delta d_k^e sin(\theta_{k-1} + \Delta\theta_k^e/2) \\ \theta_{k-1} + \Delta\theta_k^e \end{bmatrix}\end{aligned} \tag{4.6}$$

$$\mathbf{u}_k = \begin{bmatrix} \Delta d_k^e \\ \Delta\theta_k^e \end{bmatrix} \tag{4.7}$$

$$\Delta\theta = \frac{\Delta d_k^e}{L} tan(\varphi^s) \tag{4.8}$$

where Δd_k^m is the incremental distance covered in the last iteration, φ_k^s is the instantaneous steering angle of the vehicle, and L is the vehicle wheelbase.

Since the encoder sensors are publishing the raw data at a very high rate, an Extended Kalman Filter (EKF) was incorporated into the output data of the encoder odometry to filter the outliers and obtain better results.

Velodyne Odometry: The Velodyne odometry package is based on the work presented in [200], where it subscribes to the lidar pointcloud and apply almost instant lidar odometry and mapping. The package accuracy over the KITTI Benchmark is in the second place in the list of the most efficient odometry algorithms, with values of 0.64% and $0.0014^o/m$ for the mean translation error and mean orientation error respectively. Though this is considered the most accurate of all proposed localization methods, its accuracy is only effective in urban and structured environments. Therefore, the necessity of additional localization systems and a fusion is justified.

Covariance Estimation: Proprioceptive sensors measure internal states of the vehicle. For example, wheel optical encoders measure the position of the wheels, accelerometers measure the vehicle accelerations, and digital compasses or gyroscopes measure the vehicle heading angle. On the other hand, exteroceptive sensors acquire information from external observations; such as GPS modules [201]. However, proprioceptive sensors suffer from the accumulation of error in the vehicle pose estimation, which is called the drift error. It is divided into two components: deterministic and stochastic components. The deterministic component includes unequal wheel diameters, misalignment of wheels, or kinematic modeling error due to the inaccuracy of parameters measurement. However, the stochastic component occurs due to the ground condition (slippage), the temperature change, and the driving behavior [202]. Taking both components into consideration, the sensor error is modeled using Equation (4.9).

$$q^m = (1+\mu)q^t + \varepsilon \tag{4.9}$$

where q^m and q^t are true and measured variable, μ is the distance dependent scale factor, which models the measured variable dependent errors and ε models the random errors.

The deterministic component can be calibrated [203], however, the stochastic component of the error cannot be eliminated and is variant with different driving conditions. Accordingly, μq and ε are considered random variables, and it is safe to assume that these variables have zero mean Gaussian distribution $\mathcal{N}$, as shown in Equation (4.10) and (4.11).

$$\mu q^t \sim \mathcal{N}(0, \sigma_\mu^2) \tag{4.10}$$

$$\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon)) \tag{4.11}$$

Assuming zero mean errors, in case of wheel encoders the value μq^t will be μd^t and is correlated to the Standard Deviation (SD) σ_μ with the correlation constant c, as in (4.12).

$$\sigma_{mu} = c\mu q^t \tag{4.12}$$

Accordingly, the proprioceptive error model is:

$$q^m = q^t + \psi \tag{4.13}$$

$$\psi = \mu q^t + \varepsilon \tag{4.14}$$

$$\psi \sim \mathcal{N}(0, (c\mu q^t)^2 + \sigma_\varepsilon^2) \tag{4.15}$$

where σ_ε is the SD of the stochastic component of the error.

The pose of a vehicle is estimated using proprioceptive sensors based on a recursive kinematic model, which calculates the vehicle pose $\mathbf{x}_k$ using the previous pose $\mathbf{x}_{k-1}$ and the input measurement $\mathbf{u}_k$, as shown in Equation (4.16).

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k) \tag{4.16}$$

where $\mathbf{u}_k$ is the distance traveled and steering angle in case of encoders, yaw angle in case of a digital compass or a gyroscope, and accelerations in case of an accelerometer.

Given the kinematic model of the vehicle, and using Equation (4.13) for quantifying the error in a proprioceptive sensor, the generalized error model and the covariance for the inputs are presented in Equations (4.17) and (4.18) respectively.

$$\mathbf{u} = \begin{bmatrix} q_1^m \\ \vdots \\ q_n^m \end{bmatrix} = \begin{bmatrix} q_1^t + \psi^e \\ \vdots \\ q_n^t + \psi^s \end{bmatrix} \tag{4.17}$$

$$Q_u = \begin{bmatrix} (c_0\mu_0 q_0^t)^2 + \sigma_0^2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & (c_n\mu_n q_n^t)^2 + \sigma_n^2 \end{bmatrix} \tag{4.18}$$

The exteroceptive sensor measurement in the world frame $\mathcal{N}(\mathbf{Z}, \mathbf{R})$ is shown in Equation (4.19).

$$\mathbf{Z_k} = \begin{bmatrix} x_k^o \\ y_k^o \\ \theta_k^o \end{bmatrix} \text{ and } \mathbf{R} = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{y\theta} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta\theta} \end{bmatrix} \tag{4.19}$$

where x_k^o, y_k^o and θ_k^o are the observed vehicle states.

The true pose $\mathbf{x}$ most likely lies in the confidence ellipse of the measurement defined by the Eigenvalues of $\mathbf{R}$, as shown in the Figure 4.14. Hence, using the square root of the covariance matrix, a set of points in the observation distribution (sigma points) at time instance k is computed according to to [204] and shown in Equation (4.20).

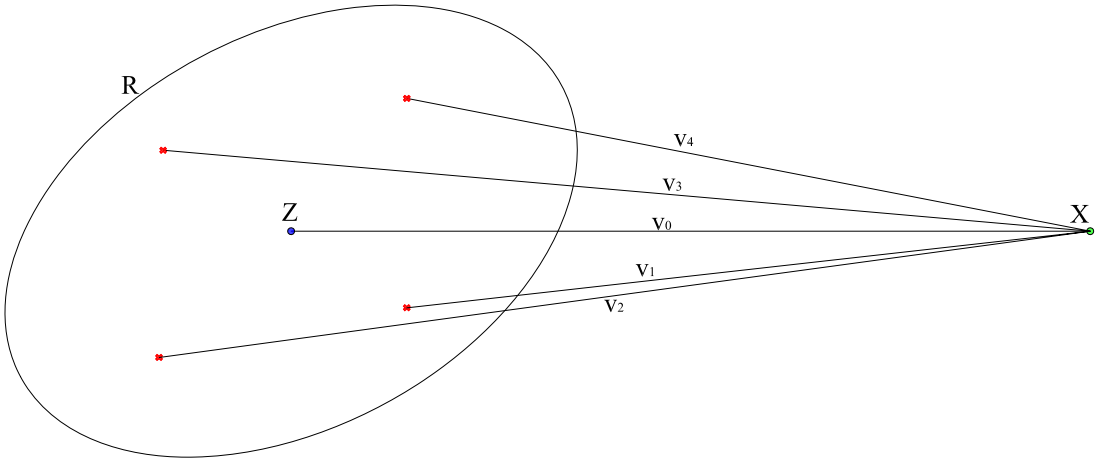


Figure 4.14 Calculating the sigma points innovation for covariance estimation

$$\xi_k = \begin{bmatrix} \mathbf{Z}_k & \mathbf{Z}_k + (\alpha\sqrt{\mathbf{R}_k}) & \mathbf{Z}_k - (\alpha\sqrt{\mathbf{R}_k}) \end{bmatrix} \tag{4.20}$$

where α is a scaling constant which depends on the accuracy of the used sensor, along with the accuracy of exteroceptive sensor covariance matrix.

In order to ensure that the vehicle pose lies inside the confidence ellipse of the exteroceptive sensor, a method for neglecting the outliers is implemented; using Mahalanobis distance threshold, as shown in Equation (4.21) and (4.22), where the estimated pose is used instead of the true pose, since the vehicle actual pose is unknown.

$$\hat{\mathbf{v}}_k^T \mathbf{R} \hat{\mathbf{v}}_k < g^2 \tag{4.21}$$

$$v_k = \hat{\mathbf{x}}_\mathbf{k} - \mathbf{Z}_\mathbf{k} \tag{4.22}$$

where g^2 is the Mahalabonis threshold and v_k is the innovation between exeroceptive and propriocetpive observations.

Using the sigma points, the sigma innovation points are sampled with Equation (4.23), and the innovation is mapped to the sensor space with Equation (4.24).

$$\mathbf{v}_\zeta = \begin{bmatrix} \mathbf{v}_{\zeta_0} & \mathbf{v}_{\zeta_1} & \dots & \mathbf{v}_{\zeta_6} \end{bmatrix} \text{ where } \mathbf{v}_{\zeta_i} = \xi_k^i - \hat{\mathbf{x}}_k \tag{4.23}$$

$$\mathbf{v}_{\zeta_i}^m = \mathbf{M}(\mathbf{v}_{\zeta_i}) \tag{4.24}$$

where $\mathbf{M}$ is a mapping function from the world frame $\mathbf{v}_{\zeta_i}$ to the sensor space $\mathbf{v}_{\zeta_i}^m$.

For example, in case of encoders, the function $\mathbf{M}$ is represented in Equation (4.25).

$$\begin{bmatrix} v_{\zeta_i}^e \\ v_{\zeta_i}^s \end{bmatrix} = \begin{bmatrix} \left\| v_{x_i} + v_{y_i} \right\|_2 \\ v_{\theta_i} \end{bmatrix} \tag{4.25}$$

After computing the innovation in the sensor space, the value is concatenated in a matrix that is referred as innovation memory matrix ξ as shown in Equation (4.26), where the choice of the size of the innovation memory matrix n should be a trade-off between accuracy in estimation and the computational cost.

$$\xi^{q_j} = \begin{bmatrix} \mathbf{v}_{\zeta_{k-n}}^e \\ \mathbf{v}_{\zeta_{k-n+1}}^e \\ \vdots \\ \mathbf{v}_{\zeta_k}^e \end{bmatrix} \tag{4.26}$$

An exteroceptive sensor does not suffer from error accumulation, but only from random error, therefore using the innovation memory matrix ξ, the measured variable dependent scale factor μ is estimated by computing the first order polynomial fit of the innovation data, using linear least squares approach, as shown in Equation (4.27).

$$\begin{bmatrix} \mu_0 \\ \hat{\mu}_{1_i}^{q_j} \end{bmatrix} = (\mathbf{K}^T \cdot \mathbf{K})\mathbf{K}^T \xi_i^{q_j} \tag{4.27}$$

where $\mathbf{K}$ is the time vector for the last n instances, $\xi_i^{q_j}$ is the i^{th} column of the innovation memory matrix of the j^{th} measured quantity, and $\hat{\mu}_{1_i}^{q_j}$ is the estimated scale factor computed from the i^{th} sigma innovation for the j^{th} measured variable q_j, as shown in Equation (4.28).

$$\hat{\mu}_1^{q_j} = \begin{bmatrix} \hat{\mu}_{1_0}^{q_j} & \dots & \hat{\mu}_{1_6}^{q_j} \end{bmatrix} \tag{4.28}$$

Accordingly, the measured variable dependent scale factor μ is estimated by calculating the weighted mean of $\hat{\mu}_1^{q_j}$, as shown in Equation (4.29).

$$\hat{\mu}^{q_j} = \sum_{i=0}^{6} W_i^{(m)} \hat{\mu}_{1_i}^{q_j} \tag{4.29}$$

The weights are adjusted as shown in Equation (4.30), however, they can be modified according to the user preference, where $W_i^{(m)}$ is weight of the i^{th} scale factor.

$$W_0^{(m)} = \frac{L}{2L+1} \text{ and } W_0^{(m)} = \frac{L+1}{4L^2+2L} \tag{4.30}$$

Therefore, using $\hat{\mu}^{q_j}$, the propriocetpive sensor covariance matrix is approximated as shown in Equation (4.31).

$$Q_u \approx \begin{bmatrix} (c_0 \hat{\mu}^0 q_0^m)^2 + \varepsilon_0^2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & (c_n \hat{\mu}^n q_n^m)^2 + \varepsilon_n^2 & \end{bmatrix} \tag{4.31}$$

where ε is a parameter representing the stochastic component of the propriocetpive sensor error, and c is adjusted according to the sensor performance.

Therefore, in order to integrate all the aforementioned localization systems into a fusion algorithm, the output of each system was passed on to the covariance estimation package first, to estimate the confidence level of the readings based on the GPS as the exteroceptive sensor.

Odometry Fusion: This package is based on the work presented in [205], where the *robot_localization* package is developed. The package is a collection of state estimation

nodes, each of which is an implementation of a nonlinear state estimator for moving vehicles. It contains two state estimation nodes, one is based on the EKF fusion algorithm, and the other is based on the Unscented Kalman Filter (UKF) fusion algorithm. The work was also extended in [168] to include another fusion approach based on Extended H-infinity filter.

Mapping

It is necessary to convert the data received from sensors to a common format, which was selected to be occupancy grid map. This mapping format allows later to perform the fusion at the cell level independently. Accordingly, a mapping library was developed for the algorithms which are presented in this work [96]. The mapping packages provide local map based on the data from the laser rangefinder, stereo camera and lidar sensors, which all are fused to have a more accurate description of the environment. Moreover, the Kinect sensor is used to create a local map for the rear-view of the vehicle based on the work presented in [206]. However, the global map is based on the lidar sensor only, which is created once per environment to be utilized for localization estimation and correction.

Perception

In this package, an architecture for dense image labeling to obtain a rich understanding of vehicle surroundings is presented, based on the work in [207]. The presented approach takes advantage of stereo information for scene segmentation. The organization of the algorithm into loosely coupled stages, which provides the proposed architecture with the capability of being extended with ease so that other classifiers can be easily integrated. The algorithm uses the stereo information in order to detect the obstacles and the free space in front of the vehicle, whereas the visible information is used to classify obstacles as pedestrians and non-traversable areas, such as gardens. Furthermore, the obstacle manager package receives the processed data from the perception system to trigger interrupters for the platform operation and take the appropriate action, either stop or maneuver.

Planning

In the planning packages, the path from the vehicle to the destination is estimated and then navigated through navigation commands.

Local and Global Planner: The local and global path planner is based on the work presented in [208]. The package creates a sequence of poses between the vehicle pose and the desired goal pose as the global plan, based on the global map of the environment. Afterward, it generates intermediates poses as the local plan, based on the local map of the environment.

It requires the velocity and acceleration limits of the vehicle, the security distance of the obstacles and the geometric, kinematic and dynamic constraints of the vehicle. One main drawback of this method is the influence of the dynamic obstacles direction in the generation of the recalculated local plan.

Path Follower: In this package, an optimal path tracking approach is proposed, which is based on Linear Quadratic Regulator (LQR). Figure 4.15 shows the vehicle dynamics over the simplified bicycle model. Applying summation to all the lateral forces yields Equation (4.32).

$$F_{yf}cos(\delta) - F_{xf}sin(\delta) + F_{yr} = m(\dot{v}_y + v_x r) \tag{4.32}$$

where F_{yf} is the lateral forces on the front wheels in the Y-axis direction, δ is the steering angle, m is the vehicle mass, v_x is the vehicle velocity in X-axis direction, and r is the angular rate about yaw axis.

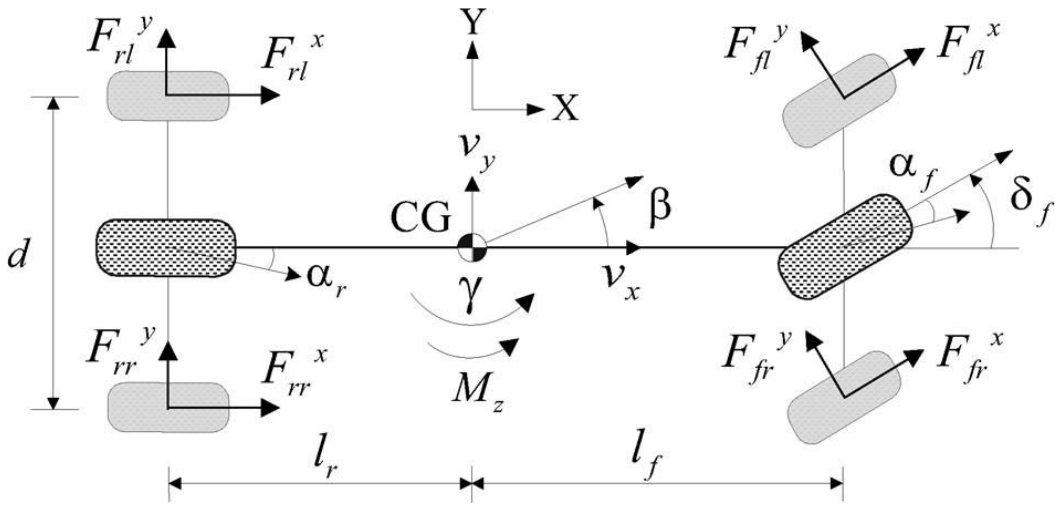


Figure 4.15 Vehicle dynamics bicycle model [209]

Based on planar driving only, the vehicle center of mass is located in the middle of the vehicle base, therefore balancing the yaw moments is estimated Equation (4.33).

$$l_f(F_{yf}cos(\delta)) - l_r(F_{yr} - F_{xf}sin(\delta)) = I_z\dot{r} \tag{4.33}$$

where l_f is the distance from the front track to the vehicle center of mass and I_z is the inertial yaw.

The slip angles of the tires are estimated as in Equation (4.34).

$$\alpha_f = tan^{-1}(\frac{v_y + l_f r}{v_x}) - \delta \text{ and } \alpha_r = tan^{-1}(\frac{v_y + l_r r}{v_x}) \tag{4.34}$$

Modeling the forces of wheels based on slip angle, as presented in [210], therefore, the lateral forces are estimated in Equation (4.35).

$$F_{yf} = -c_f \alpha_f \text{ and } F_{yr} = -c_r \alpha_r \tag{4.35}$$

Furthermore, to linearize the system, the obtained mathematical model parameters are substituted via the linear controllers applied to reduced vehicle model, thus the state space equation is formulated as:

$$\begin{bmatrix} \dot{v}_y \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{-(c_f+c_r)}{mv_x} & \frac{l_r c_r - l_f c_f}{mv_x} - v_x \\ \frac{l_r c_r - l_f c_f}{I_z v_x} & \frac{l_f^2 c_f + l_r^2 c_r}{I_z v_x} \end{bmatrix} \begin{bmatrix} v_y \\ r \end{bmatrix} + \begin{bmatrix} \frac{c_f}{m} \\ \frac{l_f c_f}{m} \end{bmatrix} \delta \tag{4.36}$$

The simplified state space model of vehicle dynamics is written as shown in Equation (4.37), which expresses the reduced vehicle modeling about the path coordinates.

$$\dot{x} = Ax + B\delta \text{ at which } x = \begin{pmatrix} e_{cg} & \dot{e}_{cg} & \theta_e & \dot{\theta}_e \end{pmatrix}^T \tag{4.37}$$

where e_{cg} is the lateral distance from the center of gravity to the path, θ_e is the heading angle deviation of the vehicle in reference to the path heading, and δ is the reference steering angle.

To apply the state feedback law, the controllability of the system is tested using MATLAB, and it is full rank. Accordingly, the feedback law is applied to the model as shown in Equation (4.38).

$$\delta = -Kx = -k1e_{cg} - k2\dot{e}_{cg} - k3\theta_e - k4\dot{\theta}_e \tag{4.38}$$

Accordingly, the eigenvalues of the $(A - BK)$ matrix are substituted at the desired location. However, to optimize the control gains, a discrete Linear Quadratic Regulator (LQR) is proposed. Firstly the system is transformed to be in a discrete-time domain, the discretization was performed using MATLAB, using a time-step of 0.0001 sec. The system state is nevertheless periodic, to eliminate lack of data and create a continuous signal to the controller, a zero order hold method was implemented.

The objective cost function was introduced to the model as a minimization problem, to place the control gains at minimum cost, which is shown in Equation (4.39).

$$J = \sum_{k=0}^{\infty} x^T(k) Q x(k) + \delta(k) R \delta(k) \tag{4.39}$$

where Q is the diagonal weighting matrix with an entry for each state corresponding to the performance aspects, and R is weighting value corresponding to the control effort.

For the selected test platform, the Q matrix was designed to penalize the change more on the relative position error and the relative rotation error, while adding less weight to their rate of change. The Q diagonal values R were experimentally adjusted to the values in Equation (4.40).

$$Q = \begin{pmatrix} 100 & 1 & 10 & 5 \end{pmatrix} \text{ and } R = 10 \tag{4.40}$$

The cost function equation was solved using the LQR design tool on MATLAB, and the proportional gains of the controller were obtained. Finally, the package inputs are the waypoints list and the vehicle pose and outputs are the steering angle and vehicle velocity.

4.5 Concluding Remarks

This chapter presented the three different platforms that are used in the work of this thesis. First, the differential mobile robot TurtleBot3, which was utilized for validation of the multiple vehicle cooperation approaches for transportation of cargo-related requests allocation and obtained results are discussed in Section 7.3. Then the automated drone SkyOnyx, which was used for validation of the generic aspect of the path planning algorithm and the heterogeneity of the cooperation algorithm 7.4. Finally, the iCab automated vehicles are presented in detail, which are the main platforms of this thesis and all the explained work was completed during the duration of the thesis with the obtained results discussed in Chapter 7.

Chapter 5

Cooperative Driving

5.1 Introduction

In this chapter[1], Section 5.2 presents the implemented communication schemes within the framework of the thesis for cooperative driving approach, each scheme purpose is explained and described its objective in the platform. Then, Section 5.3 introduces an approach for platooning of multiple vehicles based on the communication schemes and describes how it improve the vehicle environment perception. Finally, the chapter is summarized in few concluding remarks.

5.2 Communication Schemes

According to the Intelligent Transportation Systems (ITS) society definition of road entities, they are classified as vehicles, pedestrians, and infrastructure. Therefore, in order to have a cooperation among the entities, several communication schemes are required [69]. Accordingly, five vehicular communication technologies are introduced in this section, to be used in proposed automated vehicles systems, they are V2V, V2P, P2V, V2I, and I2V.

5.2.1 Communication with vehicles

Vehicular communication is pivotal to share relevant data among the vehicles in the system, this is achieved through Vehicle-to-Vehicle (V2V) communication scheme [214]. V2V communication is achieved in a number of methods, if and only if, a medium is available, for instance networks. In the literature, different approaches are researched for communication

[1] Publications of the author related to the chapter are [211–213]

with other vehicles. In reference to the available standards, such as IEEE 802.11, the proposed V2V schemes are focused on short-range handshaking connections [215, 216].

Therefore, in this thesis, an approach for inter-vehicular communication for the broad off-road environment is proposed. The approach objective is to maintain a continuous connection among the vehicles in the system. Accordingly, a Virtual Private Network (VPN) is created, which requires secure connection via the use of authentication keys and certificates. The platform connects to the VPN via any suitable internet connection using the proper authentication credentials. In reference to platform ROS software architecture, the approach utilizes the multi-master presented in [191]. This enables the platform to have a separate ROS core, thus it is self-dependent and does not operate in a centralized paradigm. Ergo, as depicted in Figure 5.1, the proposed scheme allows the platform to access two networks. One for the vehicular communication schemes, and another for any other types of communication.

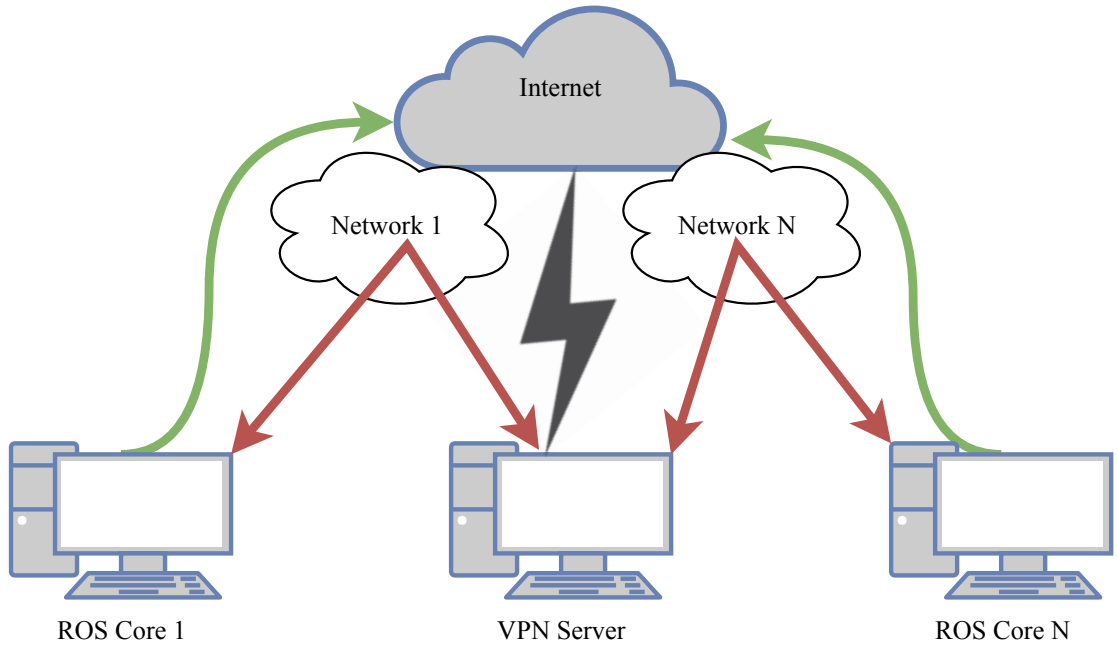


Figure 5.1 Proposed VPN architecture, green arrows utilizes the internet directly, while red arrows utilizes the virtual network

5.2.2 Communication with pedestrians

Communication with pedestrians is a crucial function for the safety of the road entities. Approaches for pedestrians safety in the Intelligent Transportation Systems (ITS) filed are divided into two categories; sensor based and communication based. Almost all available

proposed solutions are based on the earlier one [217]. However, the main disadvantage of these approaches is the dependence to be in line of sight for detecting pedestrians. Therefore, the use of V2P and P2V communications enhance the environment perception for pedestrian detection and improve the road safety [73].

Both schemes involve sending and receiving messages between the VRU and intelligent vehicles. Accordingly, a smartphone application is proposed for this purpose. The proposed application intends to reduce the risks related to the use of mobile devices in a traffic context and thereby decrease the accident exposure of pedestrians and other VRU.

Collision Prediction Algorithm

The proposed collision prediction approach is based on the algorithm presented in [218], which retrieves information from two information sources and anticipates the possible collision point. To contribute to the enhancement of the system, danger index is incorporated to the calculation using Equation 5.5. In addition to computing the collision point with reference to location coordinates, the orientation angles are estimated for both the pedestrian and the vehicle. The calculations are performed using Equations 5.1 and 5.2.

$$x_c = \frac{(y_2 - y_1) - (x_2 \cdot tan(\theta_2) - x_1 \cdot tan(\theta_1))}{(tan(\theta_1) - tan(\theta_2))} \tag{5.1}$$

$$y_c = \frac{(x_2 - x_1) - (y_2 \cdot cot(\theta_2) - y_1 \cdot cot(\theta_1))}{(cot(\theta_1) - cot(\theta_2))} \tag{5.2}$$

where x_i,y_i and θ_i are the location coordinates and heading angles of pedestrian (1), vehicle (2) and collision point (c), as shown in Figure 5.2.

The period of time to collision point (TTC) from the perspective of the pedestrian differs from the one of the vehicle. However if the difference is less than a specific threshold, the situation is considered potentially unsafe, as a collision might occur. This case is calculated in Equation 5.3, where TTC_i denotes the TTC for the pedestrian (1), the vehicle is represented by (2), (d) is the difference between them, and δ denotes the preset threshold, which determines if the road situation can be considered dangerous.

$$TTC_d = |TTC_1 - TTC_2| < \delta \tag{5.3}$$

The value of the δ parameter can be calculated based on the information of the driver reaction time t_r and braking time of the vehicle t_b, as shown in Equation 5.4.

$$\delta = t_r + t_b \tag{5.4}$$

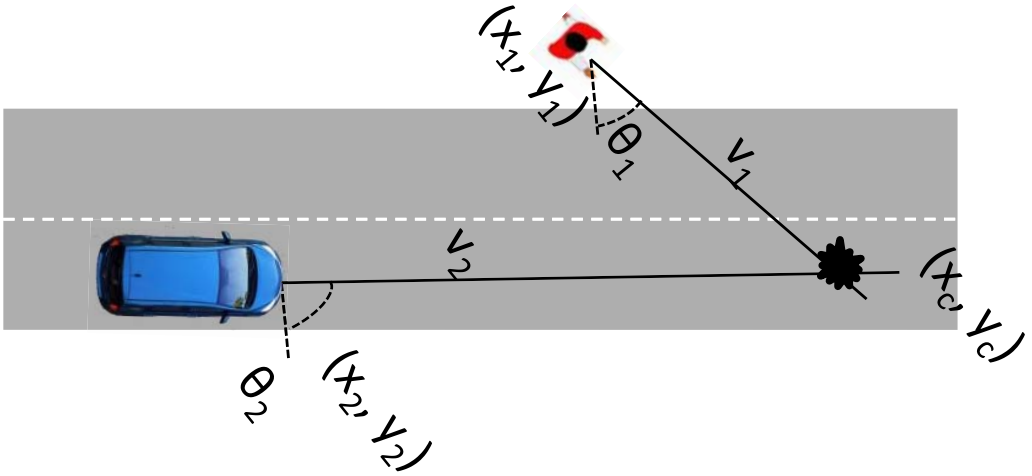


Figure 5.2 Collision prediction algorithm modeling

As the difference in the TTC is not the only factor that determines a potentially dangerous road situation, the d_i danger index was also considered in the calculations, as shown in Equation 5.5. This index is calculated for collision and no-collision trajectories, where λ is a parameter that is adjusted.

$$d_i(TTC_d, d) = \begin{cases} e^{-\frac{\lambda TTC_d^2}{2\delta^2}}, \text{for collision trajectories} \\ e^{-d\gamma}, \text{for none collision trajectories} \end{cases} \tag{5.5}$$

where the value of λ parameter is chosen to provide a danger index d_i of 0.7 When the value of TTC_d is equal to δ, the value of λ is set to 0.713. The value of γ parameter is adjusted for no-collision trajectories to ensure a specific safety distance, therefore for a desired d_i of 0.7 and safety distance of 3 meters, the value of γ is set to 0.119.

Smartphone Application

The collision prediction algorithm is implemented as a mobile application, which runs in the background when the mobile phone is active. It uses the on-board smartphone GPS localization sensor to access the pedestrian current location and uses the magnetometer sensor to access the pedestrian current orientation angle. Based on these data, the application updates the algorithm with pedestrian location information, then it waits for a nearby vehicle that broadcasts the vehicle location information.

Thereafter, the application processes the information in order to calculate and predict the location of the collision point, with respect to the pedestrian location. Moreover, it calculates

distance and time to the collision point for both pedestrian and vehicle. Last but not least, the system calculates the collision time and danger indexes to display the warning message accordingly. In the event that the application detects a situation that could jeopardize the safety for the pedestrian, it displays a warning and vibration message on the screen to inform the pedestrian about an approaching vehicle. In order to foster a quick reaction time to the message displayed, the user interface indicates the direction of the oncoming vehicle. Figure 5.3 shows an example of the calculations in the debugger and the user interface of the application.

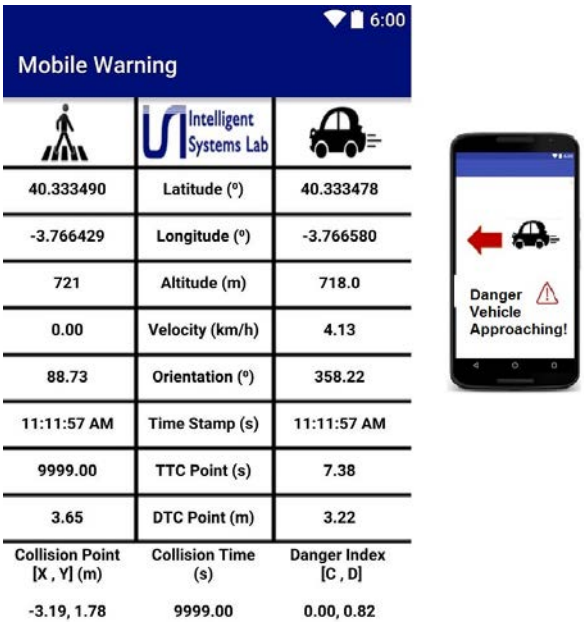


Figure 5.3 Application debug screen (left) and application user interface screen (right)

An alert control system is responsible for automatically triggering an alarm when a situation is deemed unsafe, i.e. when the values are lower than a specific threshold. This threshold is calculated in Equation 5.4, taking into account the minimum estimated reaction time of 0.66 seconds as defined in [219] and corroborated in a more recent work in [220]. When the time to collision is shorter than this value, the alarm is triggered according to the specific required reaction time determined by the type of vehicle involved and the road situation. The advantages of the proposed system are described as follows:

- Alert to foster road attentiveness, in the event that a mobile device is being used when traffic needs to be considered, the mobile application urges the user to quickly pay attention to the road and it sends a notification to the vehicle involved.

- Road safety and battery conservation. The application avoids the trigger of an alarm if the mobile phone is inactive as it would indicate that the user is not manipulating the device. At the same time it preserves the battery of the phone. Triggering an alarm in a situation that has not been classified as unsafe might provoke the opposite of the desired effect, as for example the user might turn off the system or ignore all warnings.
- Verification of reliability of automated vehicles. Information about the nearby location of an automated vehicle might help familiarize users with new technology and increase perceived safety of autonomous vehicles.

5.2.3 Communication with infrastructure

Last but not least is the communication between vehicles and the infrastructure. Nowadays, private companies advance in the development of pertinent infrastructures, in order to facilitate the next step for automated vehicles. V2I communication offers several advantages for automated vehicles, such as traffic light cycle details, potential road hazards alerts, and other contextual information. Moreover, V2I communication is essential for smart cities in many aspects, for instance, urban traffic coordination among others. Accordingly, recent collaboration between car manufacturers, such as Audi, BMW, Daimler, and Ford, in collaboration with telecom companies, such as Samsung, Verizon, SK Telecom, and Ericsson, are developing the 5G Automotive Association (5GAA) for connected mobility and road safety in the future smart cities [221].

In the proposed work, V2I communication scheme presents the infrastructure as the role of the observer, by gathering information from all vehicles in the system for monitoring and inspection purposes [222]. The infrastructure node is a webserver with a graphical user interface, which displays the system information such as vehicle location, battery level, status and tasks list, as shown in Figure 5.4.

The webserver node runs in the VPN server computer, which utilize *apache2*, *nodejs* and *mongoDb*. The node apply both V2I and I2V communication schemes. On the one hand. the V2I communication is only to monitor and observe the vehicles available in the system at all times. Thus, vehicles send their position, orientation, battery level, status, number of on-board passengers and tasks list to the webserver for display. Afterwards, the webserver shows these data over a graphical user interface for operators to keep an eye on the vehicles behavior, as shown in Figure 5.4.

On the other hand, the I2V communication has a different function, which is to register and store transport requests in a database from users in the testing environment. This request database is available for all vehicles in the system to be acquired for multi-robot cooperation

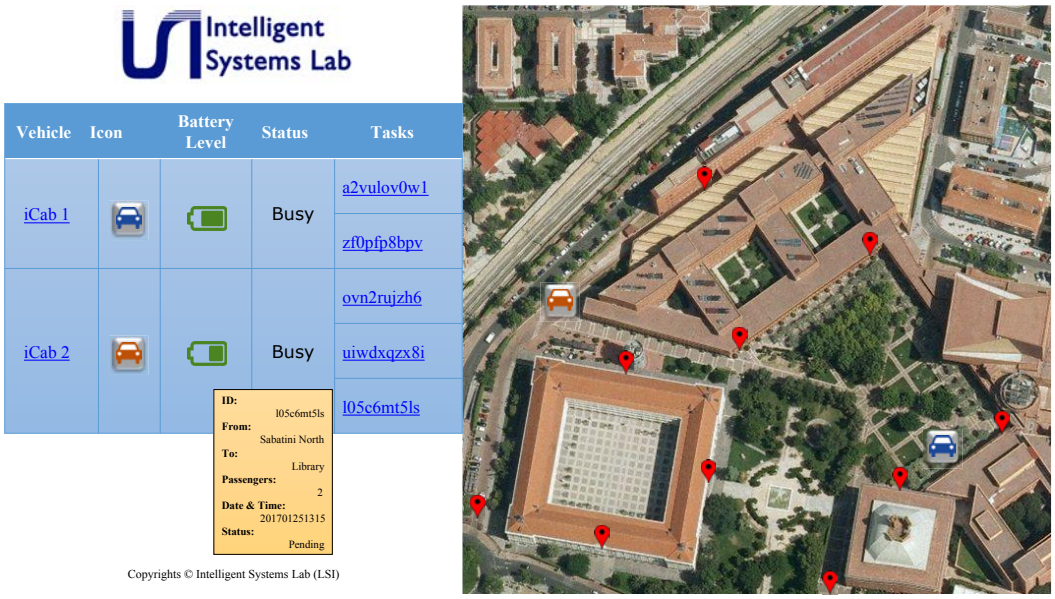


Figure 5.4 Webserver graphical user interface for V2I communication

and coordination algorithms. Therefore, the webserver is not acting as a central node, but as a hub where requests data are stored. Vehicles perform the allocation algorithm in a decentralized organizational paradigm. Through the graphical user interface, users can create transportation requests from one point to another and specify the number of passengers. One additional function of the webserver is updating the user with the status of the transportation requests via notification messages, as per the communication with the vehicles output from the allocation algorithm.

Consequently, the iCab platforms utilize all above mentioned communication schemes to apply cooperation and coordination approaches, as explained in the following sections.

5.3 Platooning Approach

In this section, a model for the cooperative vehicles platooning is described, and the algorithm with detailed steps that shows how the vehicles benefits from V2X communications to achieve an efficient cooperation between them.

5.3.1 Modeling

When people drive vehicles in a road, they take in consideration the desired path to their destination point, the surrounding vehicles position and speed, and other environment percep-

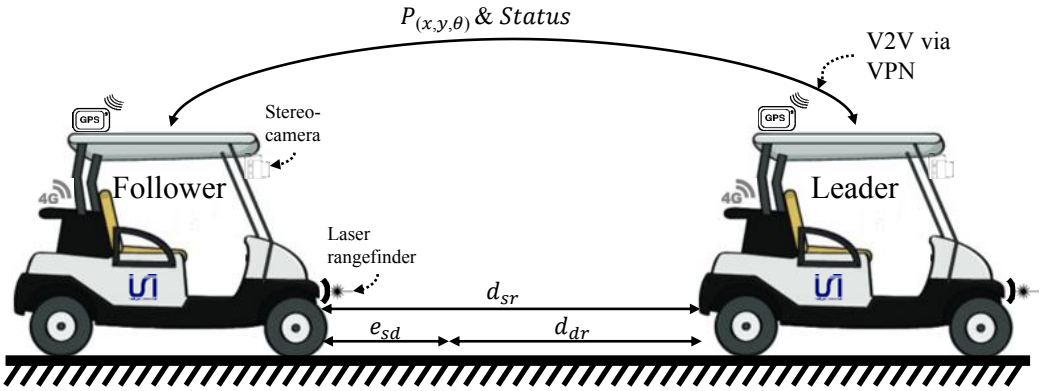


Figure 5.5 Leader-Follower platooning modeling diagram

For instance, autonomous vehicle platooning is selected to test the cooperation algorithm. A simple diagram for the leader-follower platooning model is depicted in Figure 5.5.

Accordingly the kinematics modeling that govern the motion of the autonomous vehicles can be outlined in such a way. The desired distance d_{dr} as a gap between the two vehicles is evaluated as shown in Equation (5.6).

$$d_{dr} = \frac{v_l^2 - v_f^2}{2a_{max}} + v_f.t_{br} \tag{5.6}$$

where v_l and v_f are the forward velocities of both the leader and follower vehicles respectively, a_{max} is the maximum acceleration of the follower vehicle and t_{br} is the follower vehicle braking time, which is estimated according to the vehicle braking model.

The separating distance d_{sr} is evaluated by the follower vehicle sensors. Both sensors, stereo-camera and laser rangefinder, are used to detect and classify the ahead leader vehicle [160]. Accordingly the spacing distance error e_{sd} is evaluated as per Equation (5.7).

$$e_{sd} = d_{sr} - d_{dr} \tag{5.7}$$

Along these lines, a classical proportional differential (PD) controller is used to regulate the follower vehicle velocity, as shown in Equation (5.8).

$$v_f(t) = k_p.e_{sd}(t) + k_d.\frac{d}{dt}e_{sd}(t) \tag{5.8}$$

On the other hand, the V2V communication ensures sharing the vehicles pose, position (x,y) and orientation (θ), with the each others. In addition to sharing the vehicles status, to

acknowledge which vehicle is elected to be the leader. Consequently by means of Equation (5.9), the waypoint pose with respect to the follower vehicle is estimated. Afterwards, the vehicle applies path planning algorithm to get the necessary navigation commands to reach that point.

$$\begin{bmatrix} x_p \\ y_p \\ \theta_p \end{bmatrix} = \begin{bmatrix} x_l \\ y_l \\ \theta_l \end{bmatrix} - \begin{bmatrix} sin(\theta) & 0 & 0 \\ 0 & cos(\theta) & 0 \\ 0 & 0 & 0 \end{bmatrix} .d_{dr} - \begin{bmatrix} x_f \\ y_f \\ \theta_f \end{bmatrix} \tag{5.9}$$

Utilizing the vehicles kinematics model, on-board environment perception sensors and communication schemes, the follower vehicle cooperates with the leader vehicle to follow it based on the described algorithm in the next section.

5.3.2 Algorithm

The selection of platooning mode is carried out in the reactive layer of the proposed ROS architecture. Accordingly upon the system acquires a complex task, which requires multiple vehicles to navigate the same path, it decomposes the task into simpler tasks for the reactive layer.

The platooning algorithm is divided into several steps, starting with the two vehicles navigate to the initial point of the path from their current location. Upon arriving there, each vehicle uses V2V communication to check the other vehicle pose. Thus the leadership election is achieved, according to which vehicle is ahead of the other in the direction of the initial point. Afterwards both vehicles run Algorithm 1.

This algorithm applies different functions in the system, which are described below:

- *getAheadVehicle*() function takes two poses as inputs and returns which one is ahead of the other.
- *pathPlanning*() function utilizes the optimization approach in [161].
- *getSeparationDistance*() function reads the perception sensors in allocating the leader pose utilizing [160].
- *getDesiredDistance*() function takes v_l and v_f as inputs and returns the desired distance based on (5.6).
- *getFollowerWaypoint*() function takes d_{sr} and d_{dr} as inputs and returns the waypoint pose based on (5.9).

Algorithm 1: Cooperative platooning algorithm

Input: $pose_{own}$, $pose_{other}$, $status_{platoon}$, $pose_{inital}$, $pose_{goal}$, $velocity_{own}$, $velocity_{other}$
Output: $pose_{waypoint}$, $path_{leader}$, $distance_{separation}$, $distance_{desired}$

```
status_platoon ← getAheadVehicle(pose_own, pose_other)
while pose_own != pose_goal do
    if status_platoon == LEADER then
        path_leader ← pathPlanning(pose_own, pose_goal)
        setNavigationCommands(path_leader)
    else
        distance_separation ← getSeparationDistance()
        distance_desired ← getDesiredDistance( velocity_own, velocity_other)
        pose_waypoint ← getFollowerWaypoint( distance_separation, distance_desired)
        setNavigationCommands(pose_waypoint)
    end
end
```

- *setNavigationCommands*() function takes a list of waypoints and apply the navigation approach in [161].

During the vehicles navigation, the perception system is active for obstacles detection, classification and avoidance. Moreover, the leader vehicle has the V2P and P2V communications active, in case of VRU presence, thus it receives a warning notification of possible collision point in advance. Due to the cooperative driving architecture and V2V communication, the leader vehicle shares the information of the presence of an obstacle ahead with the follower vehicle. Accordingly the necessary action of braking or maneuvering is applied in both vehicles.

The cooperative driving architecture is implemented in ROS. Therefore all shared topics and messages are inscribed with time-stamp, which allows synchronization between the vehicles systems, sensors and actuators.

5.4 Concluding Remarks

This chapter presented the various communication schemes that are implemented in the iCab platforms for cooperative driving purposes. Starting with the creation of the VPN to enable the multiple ROS core communication, then explaining how the V2V communication scheme is defined. For the V2P and P2V, a smartphone application was developed to predict possible collision point for VRU based on the positions and headings of the vehicle and the VRU. Furthermore, V2I and I2V communication schemese are presented for the purpose of system

motioning and acquiring of the transportation requests from the platform users. Finally, as a case study of cooperative driving, an approach for platooning of multiple vehicles is proposed. The modeling of the approach is dynamically based on the vehicles velocities and acceleration to maintain a distance gap in the platoon.

Chapter 6

Task Allocation

6.1 Introduction

In this chapter[1], the proposed approach flowchart for the MRTA problem is defined and detailed. Furthermore, the explanation for the proposed ROS-based generic architecture is presented. Then the chapter introduces a novel hybrid optimization-based solution to the MRTA problem. First problem formulation is introduced and the basic concepts behind this formulation are also discussed. Later, the solution construction is presented, the objective function and solution constrains. Finally, the proposed algorithm is explained and discussed.

6.2 Proposed Approach

The objective of the examined system is to collect any number of user transportation requests and allocate them to the available multiple automated vehicles. The MRTA problem governs the coordination and cooperation among the vehicles and the allocation of the requests for optimal solution. In this section, the flowchart of the proposed approach is explained, which is presented as a generic architecture for any type of transportation request, to be allocated to any type of mobile vehicle.

Figure 6.1 depicts the overview of the proposed MRTA approach flowchart. The procedures in the diagram represent the architecture continuous running loop. The loop is executed upon the connection using V2V communication scheme for sharing data among the vehicles, and the connection using V2I and I2V schemes for acquiring the transportation requests from the users and updating the requests status after allocation.

[1] Publications of the author related to the chapter are [137, 223–226]

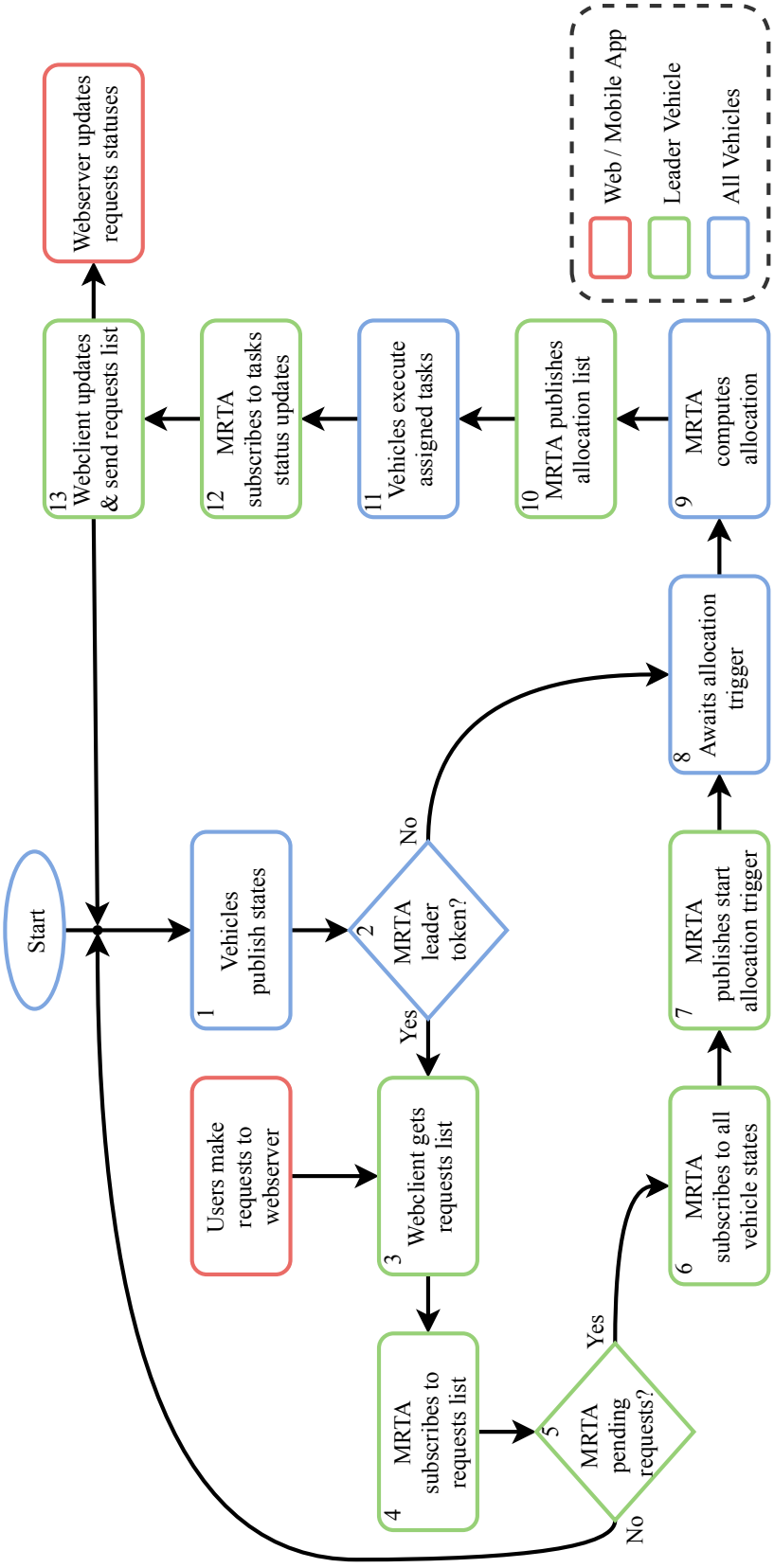


Figure 6.1 Proposed MRTA approach flowchart

The diagram consists of three types of entities for the process execution; the web / mobile application, the dynamically selected leader vehicle, and all vehicles in the system. The color code for the entities indicates the location of the process execution. The red-color is for the web / mobile application, which is managed externally and independent from the proposed approach, it governs the handling of acquiring requests from the users, and shows the requests status based on the allocation algorithm results. The green-color is for the leader vehicle, which is dynamically selected based on a selection criteria explained below. Finally, the blue-color is for all the vehicles in the system, including the leader vehicle. Moreover, the processes are numbered for easier in-text reference.

At the top of the system, users can make transportation requests using the web / mobile application, for instance the application presented in Chapter 5. Though this process is independent from the proposed approach, however it is crucial for the process flow. Once the proposed approach starts, process (1) starts and executes a trigger for all the vehicles in the system to publish their states and statuses. The published vehicles data are the global position and orientation in the environment, the battery level, the cargo capacity and the vehicle status. The vehicle statuses are:

- **Idle**: this status is set when the vehicle is static and not assigned to any request.
- **Performing**: this status is set when the vehicle is in motion with an assigned request.
- **Unknown**: this status is set in when the vehicle is neither idle nor performing.

Next, in process (2), the leader token selection algorithm is executed, which is executed in the MRTA node of all vehicles. The algorithm objective is to dynamically select the leader for the loop instance, based on the vehicles states and statuses. The leader role is to unify the inputs to the allocation algorithm and to synchronize data among all the vehicles. Though this might be considered as centralized paradigm, the leader is dynamically changing each loop. Therefore, the token selection concept ensures the decentralization of the system, at which if the leader vehicle had malfunction or loss of communication, another leader is selected to keep the whole system operating; same concept applies if a new eligible vehicle joins the system. The token algorithm checks the whole system at the beginning of every loop to ensure only one token is assigned to a unique vehicle. Accordingly, this mechanism guarantees the consistency and synchronization of the requests through the distributed system. The token selection is estimated based on weighted maximization objective function, which has the vehicle status and the battery level as inputs. The vehicle status indicates the computational power consumed by the vehicle, idle vehicle is more eligible to communicate with the webserver for the requests acquisition, same applies to the battery level.

Once this process ends and the token is assigned, the non-leader vehicle awaits the allocation trigger published by the leader in process (8), however if leader, it starts the acquisition process for the requests and other vehicle status in the following sequence. First, in process (3), the leader webclient node communicates with the webserver to get the requests list. Only the requests which are not finished are communicated to the webclient. Once requests list is available, in process (4), the leader MRTA node subscribes to it for processing. Then in process (5), the leader MRTA checks whether all requests are assigned, or there are still pending requests. In case there are no pending requests, the loop restart to await new requests for allocation. However, in case there are pending requests, the leader MRTA node subscribes to all vehicle statuses in process (6). Therefore, at this stage the allocation algorithm is ready to start, since the both requests and vehicles lists are obtained and synchronized.

Consequently, leader MRTA node triggers the start of allocation for all vehicles in process (7). In process (8), all vehicles were awaiting the allocation trigger to move forward to process (9), where all vehicles computes the allocation using the defined algorithm. This option allows the execution of a distributed computational algorithm, however if the algorithm is exactly the same in all vehicles, the process can be executed only in the leader vehicle to save computational power. Regardless the fact of the algorithm is executed in all vehicles or only in the leader vehicle, only the leader vehicle publishes the final allocation solution, in process (10) for synchronization and avoid discrepancies, because of the metaheuristic of the solution generation.

Upon the final allocation solution is published, in process (11), all vehicles start the execution process for the assigned tasks and publishes the status of the task completion back to the leader. Thus, in process (12), the leader MRTA subscribes to the tasks status updates. Accordingly, in process (13) the leader webclient node updates the requests statuses based on the tasks statuses. This process is the final step, thus upon completion, the loop restart. In the meantime, it updates the webserver with the new requests list, in order to display to the user the status of request, which can be one of the following statuses:

- **Created**: this status is set when the request is first created by the user.
- **Assigned**: this status is set when the request is assigned to specific vehicle.
- **Processing**: this status is set when the request is executed by the assigned vehicle.
- **Finished**: this status is set when the request is completed and finished.
- **Canceled**: this status is set when the request is canceled by the user.

6.3 Proposed Architecture

In this section, the software architecture for the proposed MRTA system is described. The core MRTA algorithm is connected to both the user and the vehicles. On the one hand, the connection to the user is usually implemented through a requests acquisition system. This connection is used to obtain the transportation requests and update the status of those requests. On the other hand, the connection to the vehicles is used to, first, acquire all the information about the state of the vehicles, and then command them their allocated requests. The system architecture has been designed with two key features in mind: First, the integration with the requests acquisition systems and the vehicles must be easy to perform, as long as those systems follow certain predefined rules. Second, the implementation to solve the MRTA optimization problem must be exchangeable in the algorithm. This will allow developers to focus on the problem-solving algorithm, without investing time in the software infrastructure needed to make it work in the vehicles.

The proposed system follows a ROS-based architecture, which allows an easy integration of the core MRTA algorithm with different vehicles and requests acquisition systems. This system is mainly composed of two nodes: *mrta* and *task_executor*. The *mrta* node implements the input acquisition from the requests and the vehicles, the output formatting, and the communication between vehicles. Moreover, it also implements the token-based leader selection algorithm, in order to decide which vehicle will be the leader. The leader vehicle is the one responsible to get the requests from the web server and trigger the allocation process. Finally, the MRTA problem solution is implemented as a standalone module, which can be easily replaced in the *mrta* node. Next subsections will detail the composition of the MRTA nodes and the interfaces between the proposed system and the requests and vehicle systems. On the other hand, the *task_executor* node is the connection between MRTA and the vehicle, and its main task is to send the task commands and provide their corresponding feedback.

6.3.1 Core Node

The *mrta* node has been designed in such a way so it can be configured to work with different kind of vehicles in a heterogeneous way. The configuration of each vehicle is known beforehand and includes its ID and its maximum velocity, cargo capacity and battery. Aside from this static information, the leader *mrta* node also subscribes to the state of all other vehicles in the system. By this mean, the leader node receives the position, orientation, battery level and status of each vehicle.

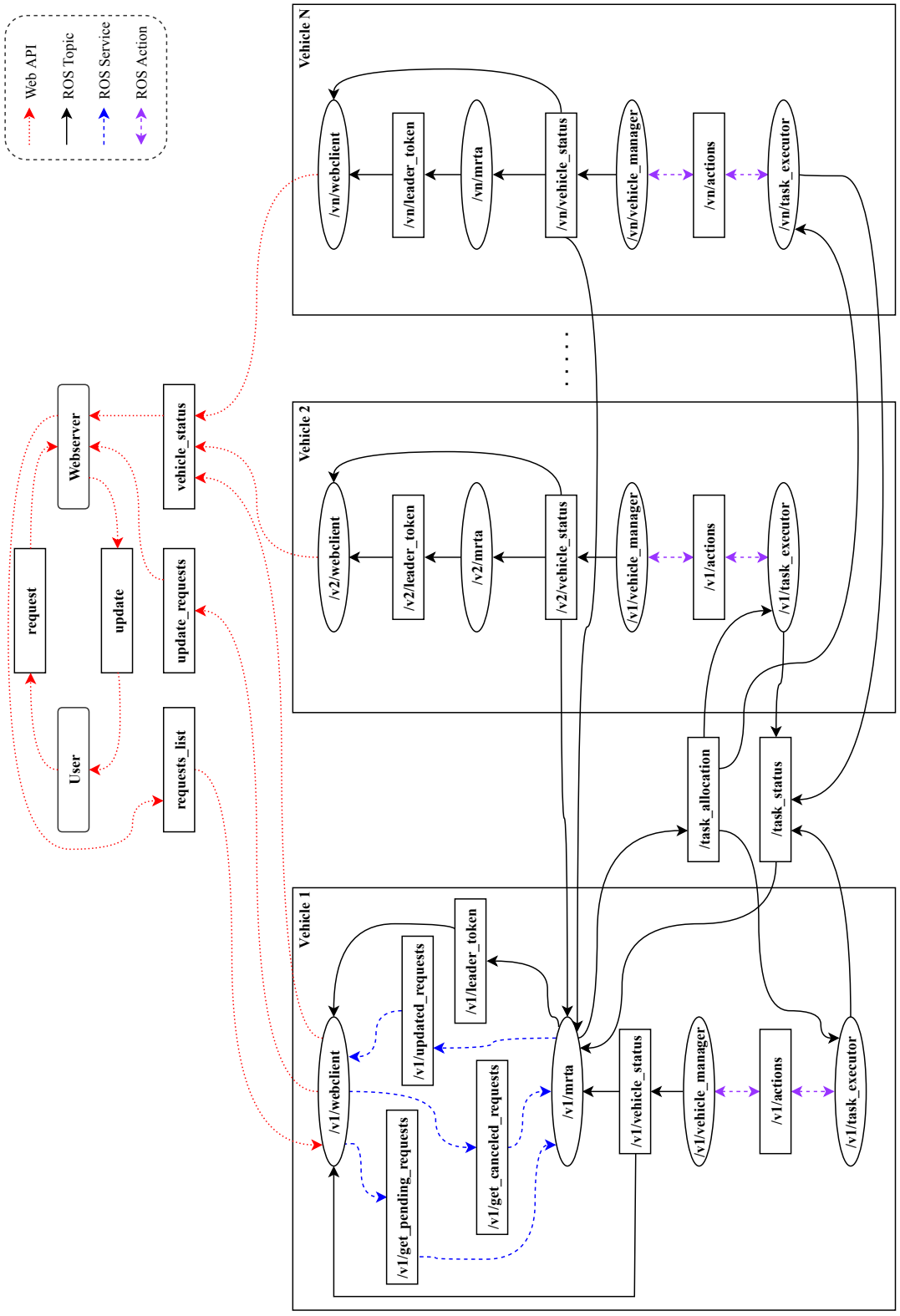


Figure 6.2 MRTA architecture

This information is sent back to all vehicles when the allocation calculation is triggered, so all the computations use the same synchronized input values. When the new set of pending requests is acquired, they are decomposed into simple tasks. The simple tasks can be of three different types:

- **Go To Task:** Basic navigation tasks containing origin and destination points.
- **Pick Up Task:** These tasks represent cargo pick up actions. They specify the location of the action and which elements must be picked up in the operation.
- **Drop Off Task:** These tasks represent cargo drop off actions. They specify the location of the action and which elements must be dropped off in the operation.

Although these are the most basic tasks for transportation purposes, new types could be added depending on the capabilities of the vehicles. The decomposition process will process each request and generate the minimum number of simple tasks needed in order to complete it. The most basic decomposition of a request will consist of a "Go To" to request origin, "Pick Up" cargo, "Go To" request destination point and "Drop Off" cargo. More complex requests may have a different decomposition, including multiple origin and destination off points. Furthermore, when the decomposition is computed, the maximum capacity of all available vehicles is taken into account; in order to split the request cargo if needed.

After the decomposition is performed, the allocation of those simple tasks is ready to be calculated, hence the trigger is sent from the leader *mrta* node to all other *mrta* nodes. The algorithm selected to do this calculation is selected by a parameter on startup, which is implemented as an external module and included in the *mrta* node. The best found solution for the MRTA problem is the one adopted, and the leader *mrta* node publishes the allocation result, so vehicles can start performing the tasks. At this point, the status of the allocated requests is updated and sent to the requests acquisition system. This status also includes the ID of the vehicles assigned to that request.

6.3.2 Requests System Side

The communication between *mrta* and the requests acquisition system is performed with ROS services. For this reason, an extra node is needed to act as a bridge between *mrta* and the requests acquisition system. In the example architecture shown in Figure 6.2, this bridge node is *webclient*. The *webclient* node is responsible of advertising the services *get_pending_requests*, *get_canceled_requests* and *updated_requests*, which are the ones needed for *mrta*. As per their names, the *get_pending_requests* service is called to get all

requests that still need to be fulfilled, the *get_canceled_requests* is called to get the latest canceled request; in order to remove them from the allocation, and the *updated_requests* service is called to send the feedback about the status of the requests back to the requests acquisition system. In the example system presented in Figure 6.2, the requests acquisition system is implemented as a Web service through a RESTful API, where the user can request a vehicle and see the status of his request. The *webclient* from the leader vehicle will send the corresponding HTTP requests to the Web API in order to get the requests and update their status. Moreover, the *webclient* nodes from all vehicles are also responsible for sending the vehicle status to the Web server, which is later used for visualization purposes.

A different requests acquisition system can be implemented, as long as the interface with the *mrta* node remains the same. For example, instead of a Web service, this system could be a desktop application, or even a random requests generator, which can be used for simulation and testing, and does not need the inputs from an actual user.

In the example system presented in Figure 6.2, the requests system is implemented as a web service through a RESTful API, where the user can request a vehicle and see the status of his request. The *webclient* from the leader vehicle will send the corresponding HTTP requests to the Web API in order to get the requests and update their status. Moreover, the *webclient* nodes from all vehicles are also responsible for sending the vehicle status to the webserver, which is later used for visualization purposes.

A different requests system can be implemented, as long as the interface with the *mrta* node remains the same. For example, instead of a Web service, this system could be a desktop application, or even a random requests generator, which can be used for simulation and testing, and does not need the inputs from an actual user.

6.3.3 Vehicles Side

The main link between *mrta* and the vehicle is the *task_executor* node. This node is responsible for parsing the allocation msg; in order to extract the next task to be performed, and also to command the execution of the tasks. Furthermore, the *task_executor* must collect the feedback about the performance of the task being executed, so it can send it back to *mrta*. Note that, unlike *mrta* and *webclient* nodes, the *task_executor* does not depend on the leader token, since its interaction with the leader *mrta* is done through global topic names, as shown in Figure 6.2.

The exact implementation of the *task_executor* node may vary depending on how the vehicle can receive the commands. In the example given, all the tasks that the vehicle is able to perform are implemented as ROS actions by the *vehicle_manager* node. This node is responsible for the actual execution of the task, and also provides feedback about the task

status through the ROS action. If the vehicle receives the commands via service or standard topic, a new *task_executor* node should be implemented with that interface. The use of ROS actions (through *actionlib*) is the recommended way to implement the task execution because it provides powerful mechanisms to control the state of the action and provide feedback about the action status.

Finally, the vehicle nodes (in the example case, the *vehicle_manager* node) are responsible for publishing the vehicle state, which includes the vehicle position, battery and status. This information is used by the MRTA algorithm; in order to compute the allocation taking into account those variables.

6.4 Problem Formulation

In this section, the MRTA problem modeling and formulation are introduced. MRTA problem [122] is formulated to allocate multiple robots, vehicles, to numerous tasks, requests. The procedures are summarized as follows:

1. Given a set of n vehicles, $V = \{\mathrm{V}_1, \mathrm{V}_2, \ldots \mathrm{V}_\mathrm{n}\}$
2. Given a set of m requests, $R = \{\mathrm{R}_1, \mathrm{R}_2, \ldots \mathrm{R}_\mathrm{m}\}$
3. Allocation of the requests to the vehicles occurs, $\mathrm{A} : R \rightarrow V$
4. Output set S is the best allocation of the requests to the vehicles

$$\begin{aligned} S = &\{\ (V_1\ R_1)\ (V_2\ R_2)\ \ldots\ (V_k\ R_k)\ \} \\ &\text{for } 1 \leq \mathrm{k} \leq \mathrm{m} \end{aligned} \tag{6.1}$$

5. Allocation S minimizes a certain objective function in order to get the best performance of the system

Accordingly, a variant of mTSP is used to model MRTA problem. In the standard mTSP formulation, n nodes are defined with the edges distances and m salesmen are known. The salesmen are required to cover all the available nodes and return back to their starting node, such that each salesman makes a round trip. The mTSP can be formally defined on a graph $G = (V,E)$ where V is the set of n nodes and E is the set of edges. Let $c = (c_{ij})$ be the distance matrix associated with E. Assuming the more general case which is an asymmetric mTSP, thus $c_{ij} \neq c_{ji}\ \forall\ (i,j) \in E$ [124]. The mTSP can be formulated as follows:

$$x_{ij} = \begin{cases} 1 & \text{if edge (i,j) is used in the tour} \\ 0 & \text{otherwise} \end{cases} \tag{6.2}$$

$$\text{minimize} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \times x_{ij} \tag{6.3}$$

$$\sum_{j=2}^{n} x_{1j} = m \tag{6.4}$$

$$\sum_{j=2}^{n} x_{j1} = m \tag{6.5}$$

$$\sum_{i=1}^{n} x_{ij} = 1, j = 2, ..., n \tag{6.6}$$

$$\sum_{j=1}^{n} x_{ij} = 1, i = 2, ..., n \tag{6.7}$$

$$x_{ij} \in \{0,1\}, \forall (i,j) \in E \tag{6.8}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |SubTour| - 1, \forall SubTour \subseteq V \backslash \{1\}, SubTour \neq \phi \tag{6.9}$$

where Equation (6.3) represents the objective function which is the summation of the total distance traveled, Equations (6.4) and (6.5) ensure that exactly m salesmen departed their starting node and returned back. Equations (6.6), (6.7) and (6.8) are the usual assignment constraints. Finally, (6.9) is the subtour elimination constraint.

The proposed formulation is extended and adapted to the problem, where vehicles represent the salesmen and requests represent the cities. Therefore, vehicles capabilities and requests requirements are considered and included in the mTSP implementation. The added features of the vehicles are capacity, velocity, energy, efficiency and sensors, and the ones for the requests are timestamp, priority, and passengers.

The vehicle capacity is the maximum number of passengers that it can hold, the velocity is a representation of the maximum speed it can reach, the energy is a representation of the battery level, the efficiency is a representation of the aging factor, and finally the sensors are a set of on-board devices to consider the vehicles as heterogeneous robots. On the other hand, the request timestamp is the date and time of the request creation, the priority is a representation or the request urgency based on the request type, and passengers is a representation of the number of users for the request.

6.4.1 Solution Construction

The solution is constructed as a set that includes a list of all vehicles in the system, followed by their assigned requests. The order of the list defines the quality of the solution according to the objective function. For example, a problem with three vehicles and five requests, one of the possible candidate solutions is represented as follows:

$$\text{Candidate Solution} = [V1\ R1\ R2\ V2\ R3\ R4\ V3\ R5] \tag{6.10}$$

Any combination of this list presents another candidate solution, and since the mTSP is a permutation problem, therefore the order of this list affects the quality of the solution as per the objective function. The order implies that each vehicle is going to execute all requests that succeeds it. In the candidate solution presented in Equation (6.10), requests 1 and 2 are executed by vehicle 1, requests 3 and 4 are executed by vehicle 2, and finally request 5 is executed by vehicle 3.

6.4.2 Objective Function

Although the MRTA problem is formulated as an instance of the mTSP, however the same objective function of the mTSP previously explained in Equation (6.3) cannot be straightforwardly used as the objective function for the MRTA problem. Therefore, some variations had to be introduced to the objective function of the mTSP in order to be effectively used for the MRTA problem [227].

There is three main variation of the MRTA problem objective function than the mTSP objective function. First, it is a multi-objective function instead of a single objective function, second, the variable to be minimized is the time rather than the distance, and third minimizing the time of the maximum subtour rather than minimizing the total time, thus dealing with it as a MinMax problem. Then for k subtours and r requests in each subtour, the total traveling time is calculated as follows:

$$\begin{aligned} \mathrm{f}(\mathbf{x}) = & \underset{j\in\{1,2,\ldots k\}}{\arg\max} \frac{\sum_{i=1}^{r-1} \text{distance}(\ subtour_{j_i}, subtour_{j_{i+1}}\ )}{\text{vehicle velocity}_j} \\ & + \\ & \underset{j\in\{1,2,\ldots k\}}{\arg\max} \frac{\sum_{i=1}^{r} \text{execution time}\ (\ subtour_{j_i}\ )}{\text{vehicle efficiency}_j} \end{aligned} \tag{6.11}$$

6.4.3 Solution Constraints

Although any arrangements of the vehicles and requests solution set are considered as a candidate solution for the mTSP problem, however, this does not guarantee that this solution is feasible for the MRTA problem. Therefore, few constraints are applied to the obtained solution, to ensure its validity and feasibility, which means checking whether the vehicle capabilities and request requirements are matching.

The first constraint is related to the capacity, for example, a vehicle with a maximum capacity of 4 passengers cannot handle a request of 6 passengers in one go, thus the request is decomposed into several requests and re-allocated accordingly. The next constraint is the energy, the vehicle battery level is always taken into consideration before the final allocation of the request, since if the vehicle does not have sufficient energy for a specific request, it is re-allocated to another vehicle. Another constraint is the priority level of the request, which implies that some requests, maintenance request for instance, must be executed first. Last but not least, a constraint related to the mounted on-board sensors, which check if the request requires the presence of a specific sensor in the vehicle.

These applied constraints strongly affect the search space of the problem through decreasing the number of candidate solutions that can be accepted as feasible solutions. One may think that this decrease of the number of feasible solutions among the candidate solutions may make it easier for the applied algorithm to find the best solution than the case without the constraints. However, these applied constraints, in fact, make it more difficult and more time consuming to find the best solution. This is mainly because the algorithm will be visiting a large number of solutions that are candidate solutions of the mTSP, but are not feasible to solve the MRTA problem.

6.5 Proposed Algorithm

The proposed algorithm is designed as metaheuristic optimization approach. It is a hybrid approach, which is based on both trajectory-based and population-based techniques. The trajectory-based is the family of optimization techniques that use a single solution throughout the algorithm, in order to find the near-optimal solution. While the population-based is the family of optimization techniques that iteratively transforms a set of solutions, in order to generate a new population of solutions with the aim of finding the near-optimal solution [224].

On the one hand, Simulated Annealing (SA) was selected as a trajectory-based approach, where the neighboring operator is randomly chosen at each iteration for diversity. The mutation operators are swapping, deletion and insertion, inversion, and scrambling. On the other hand, Genetic Algorithm (GA) was selected as the population-based approach, where

the selected mutation operators are the same, however, additional crossover operators are selected, which are partially mapped crossover and order crossover.

The swapping operator chooses two random elements of the solution list and swaps them with each other. Deletion and insertion operator chooses a random element and delete it from its current position and randomly insert it in a new position. The inversion operator chooses two random positions and inverts the order of elements between these two positions. The scrambling operator picks two random positions and scrambles the elements between these two positions. Figure 6.3 illustrates an example of all proposed operators, where positions 1 and 6 are selected as the two random elements. At each iteration, one of the four mutators is randomly chosen in order to generate a neighboring solution of the current solution. The four methods vary in their diversification and intensification effect on the generated neighboring solution.

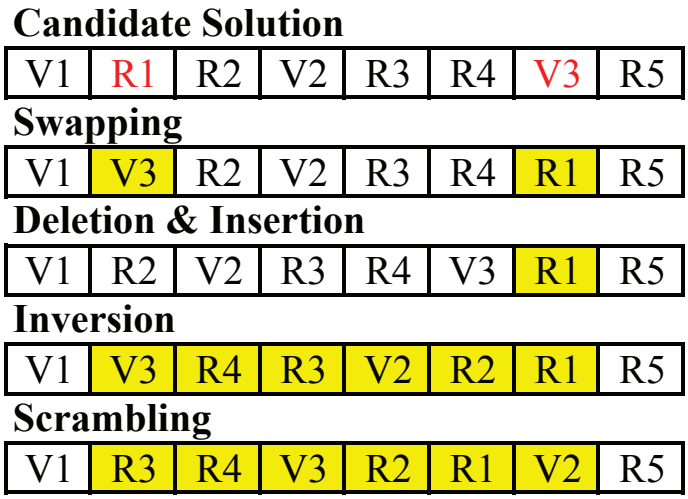


Figure 6.3 Example for the mutation operators

The crossover operator is the mimicking of the biological recombination between chromosomes, when some portion of the genetic material is swapped between chromosomes producing a new offspring chromosome. On the one hand, in the partially mapped crossover, two points are selected at random in both parents solutions and the offspring is created by exchanging the in-between chromosomes. On the other hand, in the order crossover a portion of one parent is mapped to a portion of the other parent, then from the replaced portion on, the rest is filled up by the remaining genes, where already present genes are omitted and the order is preserved. Figure 6.4 illustrates an example of proposed crossover operators, where positions 2 and 5 are selected as the two random elements. At each iteration, one of the two crossover mutators is randomly chosen in order to generate a neighboring solution of the current solution. Additionally, the random choice of the used operator gives the algorithm

both the explorative and exploitative features, that are useful in escaping local minimum and finding a better solution through searching in the neighbors of elite solutions respectively.

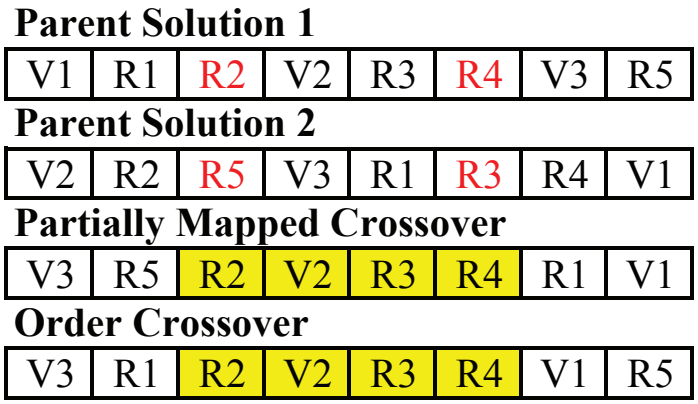


Figure 6.4 Example for the crossover operators

The algorithm is used to solve the formulated model of the MRTA problem. Inputs are the list of the requests with their requirements, the list of vehicles with their capabilities and the matrix of the distances between the points of interests of the requests locations. At each iteration of the algorithm, a solution is constructed to be evaluated, initially, it is random. The initial solution set is always filled with random elements of the requests list, and the vehicles list until both lists are empty; the only constraint is that the start of the solution must be an element of the vehicles list. After the construction of the initial candidate solution or any other neighboring solution through the algorithm iterations, the feasibility of this solution must be checked, according to the solutions constraints. As the algorithm progress, neighboring solutions of the current solution must be generated in order to explore the search space of the problem. Algorithm 2 presents the proposed algorithm used to solve the MRTA problem in this paper.

Where the parameters can be defined as follows:

- *parentsList* is the list with parents solutions
- *childrenList* is the list with children solutions
- *nextGenerationList* is the list with next generation solutions
- *iterationsNumber* is the number of iterations
- *elitismPercent* is the elitism percent
- *populationSize* is the population size
- *initialTemperature* is the initial temperature

Algorithm 2: Proposed hybrid optimization-based algorithm

Input: Requests list *requests*, Vehicles list *vehicles*, Distances matrix *distances*
Output: Best allocation *bestAllocation*

```
1  begin
2      for i ← 1 to populationSize do
3          parentsList ← generateValidSolution(requests, vehicles, distances)
4      end
5      currentAllocation ← minimumOf(parentsList)
6      for i ← 1 to iterationsNumber do
7          if i ≤ 25% of iterationsNumber then
8              elitismPercent = 20%
9          else if i > 25% of iterationsNumber AND i ≤ 50% of iterationsNumber then
10             elitismPercent = 30%
11         else if i > 50% of iterationsNumber AND i ≤ 75% of iterationsNumber then
12             elitismPercent = 40%
13         else
14             elitismPercent = 50%
15         end
16         childrenList ← crossover(least 20% of parentsList)
17         childrenList ← mutation(top 80% of parentsList)
18         nextGenerationList ← minimum elitismPercent of parentsList
19         nextGenerationList ← minimum elitismPercent of childrenList
20         for j ← 1 to (100% − elitismPercent × 2) of iterationsNumber do
21             currentAllocation ← generateValidSolution(requests, vehicles, distances)
22             currentCost ← getAllocationCost(currentAllocation)
23             bestCost ← currentCost
24             (currentAllocation, currentCost) ← temperatureLoop() // Algorithm 3
25             nextGenerationList ← currentAllocation
26         end
27         if minimumOf(parentsList) < currentAllocation then
28             currentAllocation ← minimumOf(parentsList)
29         end
30         parentsList ← nextGenerationList
31     end
32     bestAllocation ← currentAllocation
33 end
```

- $finalTemperature$ is the final temperature
- $currentTemperature$ is the current temperature
- $iterationsPerTemperature$ is the number of iterations per temperature decrements
- α is the geometric coefficient

Algorithm 3: SA temperature loop

Input: Requests list *requests*, Vehicles list *vehicles*, Distances matrix *distances*
Output: Current allocation *currentAllocation*, Current cost *currentCost*

```
1  begin
2      while currentTemperature < finalTemperature do
3          for i ← 1 to iterationsPerTemperature do
4              neighborAllocation ← generateNeighborSolution(currentAllocation)
5              neighborCost ← getAllocationCost(neighborAllocation)
6              if neighborCost < currentCost then
7                  currentAllocation ← neighborAllocation
8                  currentCost ← neighborCost
9                  if neighborCost < bestCost then
10                     bestAllocation ← neighborAllocation
11                     bestCost ← neighborCost
12                 end
13             else
14                 Generate: random number randomNumber ∈ ]0,1[
15                 transitionProbability = exp(-(neighborCost - currentCost) / currentTemperature)
16                 if transitionProbability > randomNumber then
17                     currentAllocation ← neighborAllocation
18                     currentCost ← neighborCost
19                 end
20             end
21         end
22         currentTemperature = currentTemperature * α
23     end
24 end
```

- *transitionProbability* is the transition probability
- *currentAllocation* is the current allocation solution
- *currentCost* is the current solution cost
- *neighborAllocation* is the neighbor allocation solution
- *neighborCost* is the neighbor solution cost
- *bestAllocation* is the best allocation solution
- *bestCost* is the best solution cost

6.6 Concluding Remarks

This chapter presented the detailed process flowchart of the proposed MRTA problem approach for allocation of transportation requests to automated vehicles. The approach is easily extended to any type of transportation and any type of vehicle, which provides a generic aspect and adaptability. Moreover, the approach is independent from requests acquiring system. Furthermore, the ROS-based architecture for the MRTA problem is explained. Each element of the architecture was explained and described its role in the overall system. Furthermore this chapter introduced and discussed the extension of mTSP formulation for MRTA problem. The solution construction method, the objective function and solution constraints of the problem were presented. Finally a hybrid optimization-based algorithm, using SA and GA was proposed to solve the MRTA problem.

Chapter 7

Results and Discussion

7.1 Introduction

In this chapter, the results from all the carried-out work in the thesis are discussed. Section 7.2 presents the simulator validation results, then Section 7.3 presents the results of the coordination architecture using the TurtleBot3. Section 7.4 presents the results of the planning algorithm using SkyOnyx. Sections 7.5, 7.6, 7.7, and 7.8 discuss the results of the different implemented approaches in iCab platform. Afterwards, the results of the proposed solution method to the MRTA problem are compared to well-known benchmarks and analyzed in Section 7.9. The majority of the work in this chapter are published in peer-review publications, as per the list in Appendix A.

7.2 3DCoAutoSim Validation Results

7.2.1 Setup

The simulator is implemented on a computer with the specifications listed in Table 7.1.

Table 7.1 Simulator computer specifications

Processor	8 Cores, 8 Threads @3.5GHz
Memory	16GB DDR4-2133MHz
Graphics	GTX 970M, 3GB GDDR5

In order to control the vehicle in the simulator, a Thrustmaster T500 RS controller is used as the steering wheel, in addition to its throttle, brake and clutch pedals, and the TH8 RS gear shifter, as shown in Figure 7.1.

(a) Pedals (b) Gears shifter (c) Steering wheel

Figure 7.1 Simulator controllers

The controller is connected to a car play seat, shown in Figure 7.2, to simulate a real vehicle and the simulator visuals are displayed using overhead HD beamer with resolution of $1400x1050$, in addition to a five point one surround sound system. Detailed description is presented in the ReadMe in Appendix C.

Figure 7.2 3DCoAutoSim driving simulator car seat

7.2.2 Scenario

The selected scenario was the surroundings of the University of Applied Sciences Technikum Wien, in the city of Vienna. The trajectory was a total distance of 2.6 *km*, which included intersection, traffic lights, a roundabout and pedestrians crossing. The route was defined over OSM, as shown in Figure 7.3, which is considered as the theoretical path in all experiments. The theoretical path is represented as the waypoints, such that the center of the vehicle should be in the center line of the road lane.

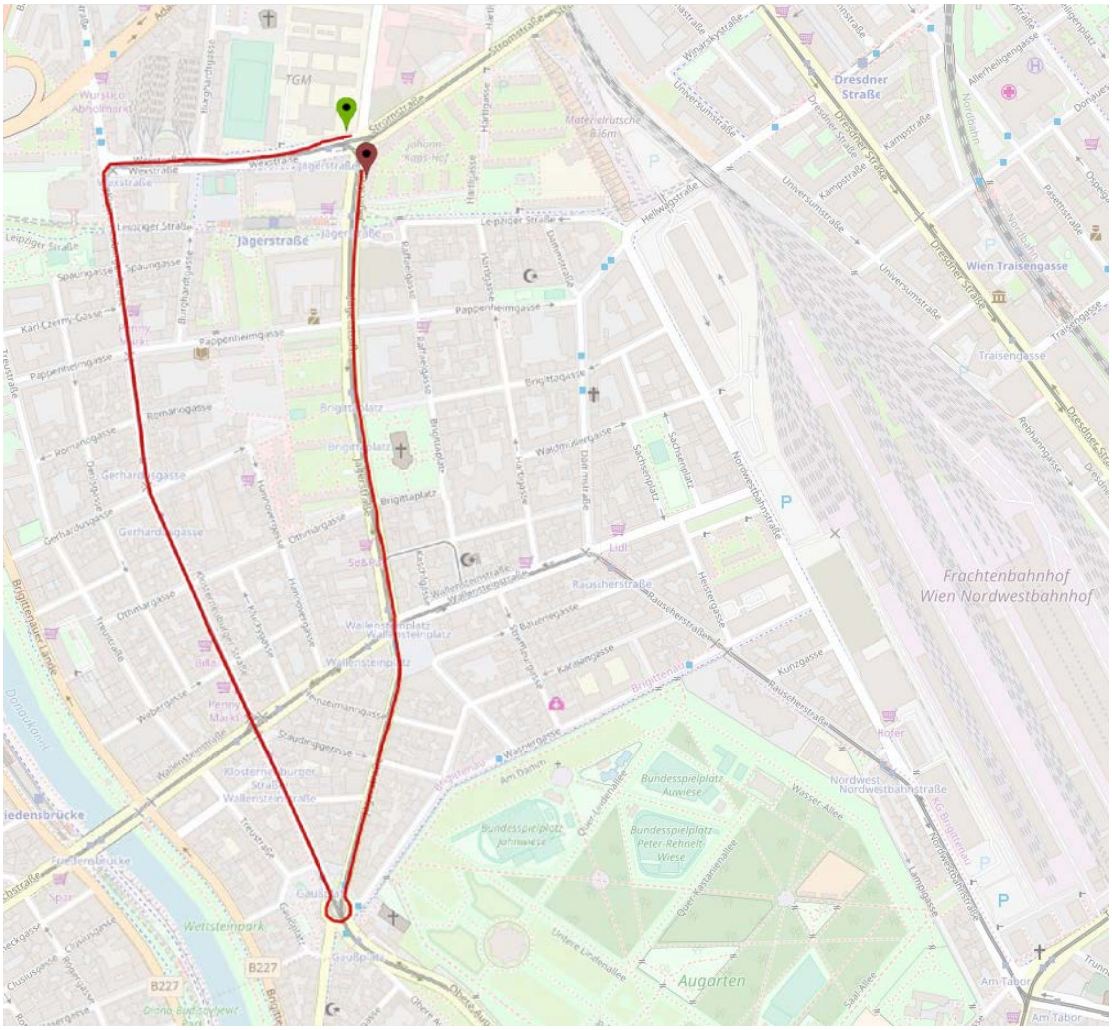


Figure 7.3 Selected path over the OSM of Vienna, where the green pin is the starting point and the red pin is the ending point

Multiple experiments were carried-out for the selected scenario, which are summarized as follows:

- **Driver Real Car**, users drive the selected route using a car equipped with on-board OBD-II transmitter. The recorded data are the vehicle GPS coordinates, orientation, velocity, acceleration, and CAN bus data.
- **Driver Simulator**, users drive through the selected route using a car in the simulator, same data parameters are recorded for later comparison.
- **Automated Simulator**, the simulator uses the route waypoints and the path tracking methods of Unity to navigate the same car on its own. Same data parameters are recorded for later comparison.
- **Automated ROS**, the simulator connects to ROS, where optimal path tracking nodes use the route waypoints to navigate the same car on its own. Same data parameters are recorded for later comparison.

7.2.3 Metrics

In order to evaluate the functionality and efficiency of the simulator, evaluation metrics are calculated for the vehicle position. Accordingly, the mean and maximum relative position error percentages are calculated as shown in Equations (7.1) and (7.2) respectively.

$$PE_{mean}[\%] = \frac{\frac{1}{N}\sum_{k=1}^{N}\left\|\begin{bmatrix}\hat{x}_k\\ \hat{y}_k\end{bmatrix} - \begin{bmatrix}x_k\\ y_k\end{bmatrix}\right\|_2}{TotalDistance} \tag{7.1}$$

$$PE_{max}[\%] = \frac{\frac{1}{N}max(\left\|\begin{bmatrix}\hat{x}_k\\ \hat{y}_k\end{bmatrix} - \begin{bmatrix}x_k\\ y_k\end{bmatrix}\right\|_2)}{TotalDistance} \tag{7.2}$$

where $\hat{x}_k, \hat{y}_k$ are the coordinates of the vehicle and x_k, y_k are the theoretical coordinates of the vehicle at time step k.

The metrics represent the vehicle lateral deviation from the center of the lane in which the vehicle drives. They characterize driving performance and performance degradation. The lane position was measured as the distance between the vehicle center and the lane center and depends on the lane geometry. Coordinates re-sampling and interpolation were implemented to calculate the lateral path deviation for all the vehicles that were driving at different velocities. To this end, each point in the experiment path was orthogonally projected to the lane center. The standard deviation of lateral position (SDLP) is calculated as the root mean square of the error.

7.2.4 Qualitative and Quantitative Analysis

Figure 7.4 depicts the followed trajectory by the vehicle for each carried-out experiments, where the green point represents the starting position, and the red point represents the final position. The compared trajectories correspond to the four experiments (Driver Real Car, Driver Simulator, Automated Simulator and Automated ROS), in addition to the theoretical path, which is the waypoints of the path shown in Figure 7.3.

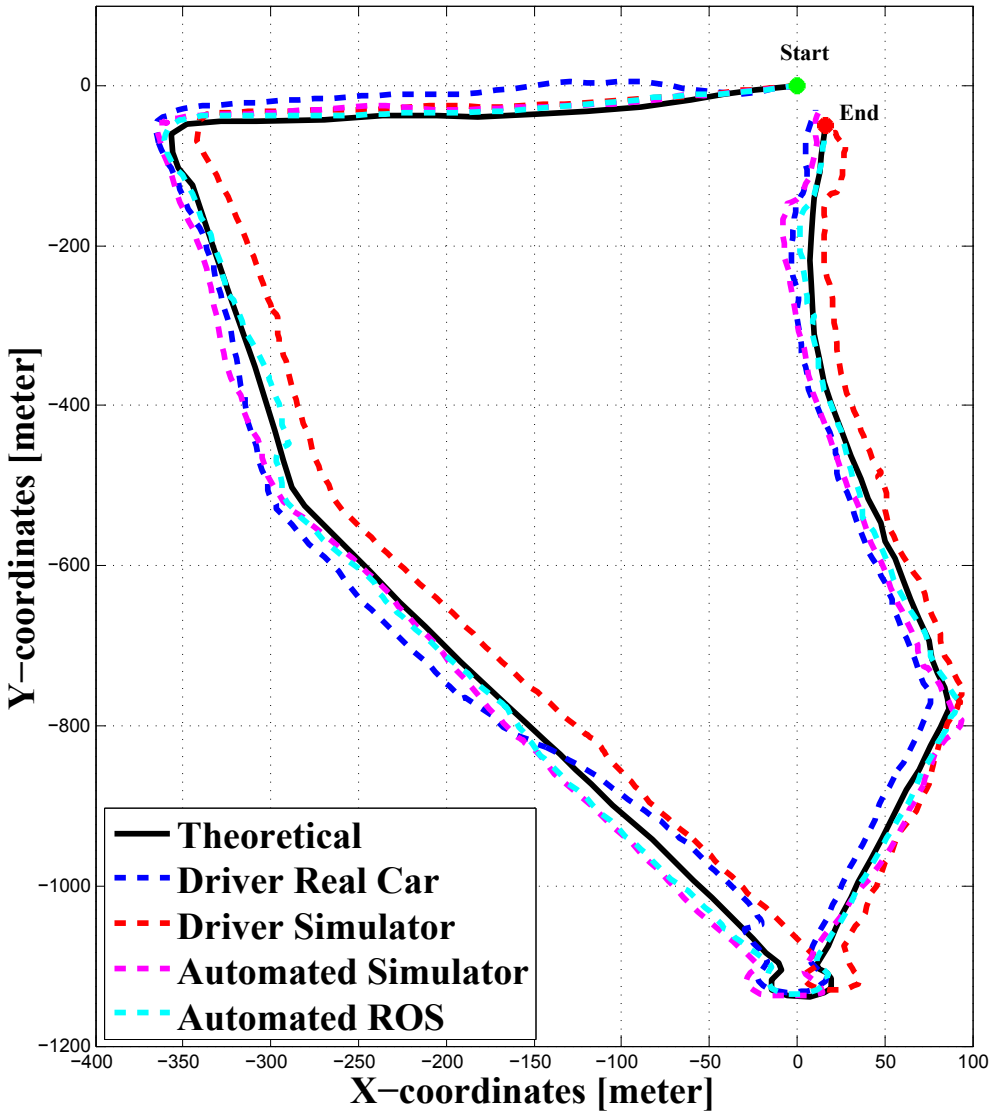


Figure 7.4 Paths comparison against the theoretical one

The automated simulator experiment using ROS path tracking delivered the best results in terms of deviation from the lane center. This was due to the fact that it relied on the

Ackermann modeling for the simulation of kinematic and dynamic parameters. The driver real car and automated simulator experiments delivered similar results in terms of smoothness and overall error. The driver simulator experiment delivered, however, a constant lateral-error during a period of 13 minutes on average per driver.

Out of 40 driving experiments, Table 7.2 summarizes quantitative results for the trajectories shown in the aforementioned section. The obtained results emphasize the qualitative analysis, where the automated driving using ROS obtained PE_{mean} of 0.38% (SD: 0.21%), followed by the automated driving using Unity with PE_{mean} of 0.64% (SD: 0.28%). The manual driving of the cars in both real and simulator obtained less accurate results, due to the fact that human error is involved. Driving the simulator car obtained the maximum PE_{mean} of 1.01% (SD: 0.48%).

Table 7.2 Quantitative results for the simulator validation

Metrics	PE_{mean} **[%]**	PE_{max} **[m]**	v_{mean} **[m/s]**
Driver Real Car	0.88%	37.692	4.79
Driver Simulator	1.01%	25.523	5.69
Automated Simulator	0.64%	16.879	5.11
Automated ROS	0.38%	13.421	4.89

The results show the viability of the proposed simulator to emulate ideal conditions, which are defined for automated vehicles and then compared with manual driving in a real test field or in a simulator environment. The simulator functionality and efficiency were validated by the performed experiments. The performance decrease shown in the driver simulator experiment was due to the settings difference, in terms of velocity in the simulator.

7.3 TurtleBot3 Results

In this section, several experiments were carried-out using GAZEBO simulator, in which three TurtleBot3 platforms were selected to validate the proposed cooperation and coordination architecture in Chapter 6.

7.3.1 Scenario

Figure 7.5 shows the designed environment of dimensions of 6×9 meters. The red-filled circles simulate the possible target locations to reach by each of the platforms.

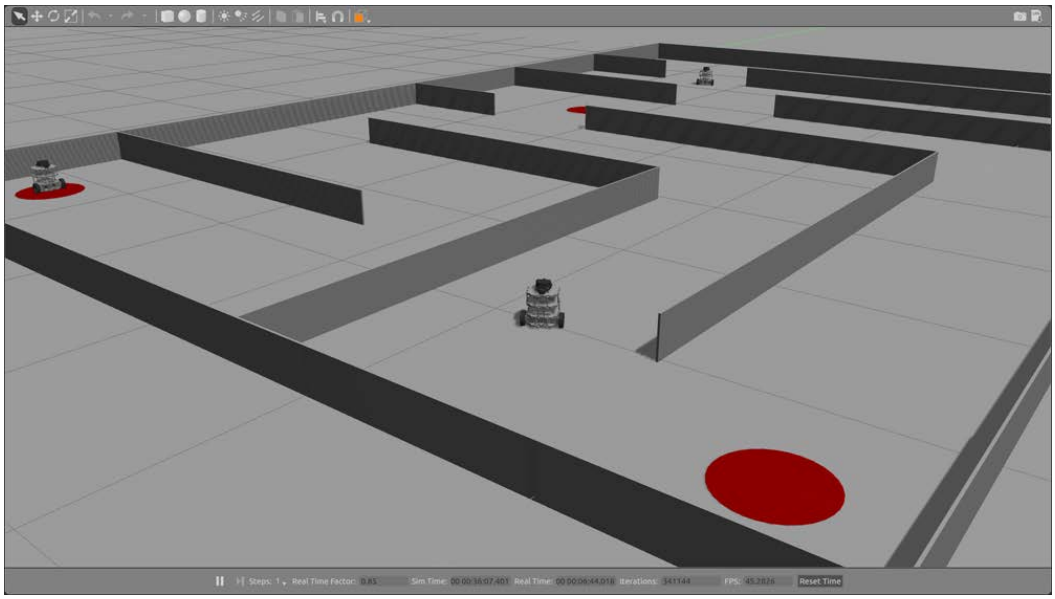


Figure 7.5 TurtleBot3 platforms simulated environment in GAZEBO

Accordingly, the environment is formulated using mTSP for the MRTA problem, where the tasks have instantaneous arrival completion. As previously mentioned in Chapter 6, the inputs are the vehicles list, the tasks list, and the distance matrix between the tasks locations. Each vehicle and each task has been previously defined with the features and requirements, therefore it is a must to consider the features of the vehicles and the requirements of the tasks as inputs to the experiment as well.

7.3.2 Qualitative and Quantitative Analysis

Figure 7.6 shows the ROS visualization of the TurtleBot3 platforms in the simulated environment. The platforms performed SLAM to generate the static map of the environment using the on-board lidar scan, which is represented with red points. The five tasks and three platforms represent a small-scaled problem for the validation purposes of the proposed architecture. After executing the solving algorithm presented in Chapter 6, the obtained results are reported in Table 7.3. These results validate the viability of the proposed approach in its integration with ROS-based platforms, before the deployment in the iCab platforms.

Table 7.3 TurtleBot3 allocation results

Best Allocation Cost [m]	**Average Comp. Time [s]**	**Deviation Error [%]**
12.76	1.04	3%

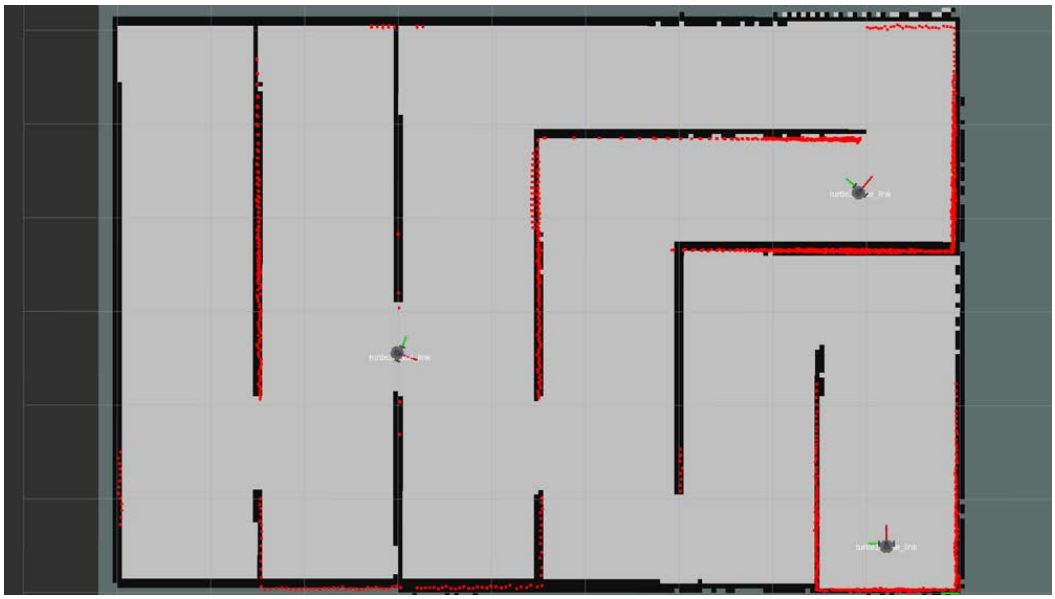


Figure 7.6 TurtleBot3 platforms environment perception in RVIZ

7.4 SkyOnyx Results

In this section, the planning approach for the UAV, which is presented in Chapter 4, is validated through several scenarios and experiments.

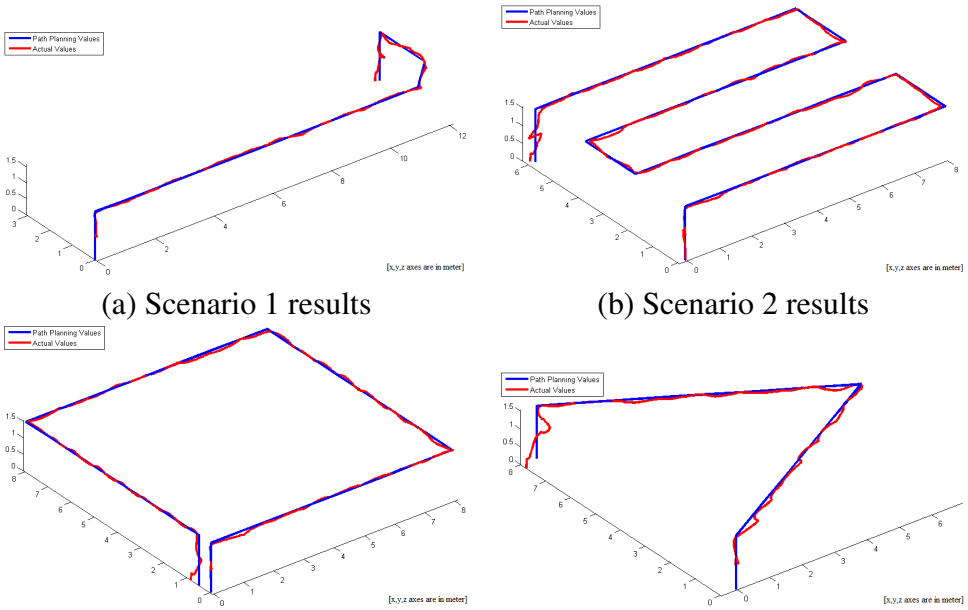
7.4.1 Scenarios

Four different indoors scenarios were tested to evaluate the performance of the proposed approaches on the platform. The scenarios were implemented in an indoor field of 9×18 meters. Each scenario was selected to analyze the planning and the control of the platform in different situations and constraints. The occupancy grid maps for the four scenarios were generated to the scale of the real dimensions, the grid cells are divided into 1 square meter. All the static obstacles are considered in the mapping process. Subsequently the maps are loaded into the system to start the path planning algorithm and to obtain the final 3D waypoints for the automated navigation.

7.4.2 Qualitative and Quantitative Analysis

Results from the four experiments are evaluated qualitatively and quantitatively. The different proposed scenarios were used to test the algorithms qualitatively. However, in order to be able to quantitatively test proposed algorithms, two evaluation metrics were introduced. First one

is the time taken by the UAV to navigate from the starting point to the goal point, including the take-off and landing duration. The second one is the error margin between the theoretical and actual paths. The performance of each scenario is illustrated through plotting the values of theoretical path planning algorithm against the localization path; where the axes refer to the real 3D coordinates of X, Y and Z-axes in meters.



(a) Scenario 1 results (b) Scenario 2 results

(c) Scenario 3 results (d) Scenario 4 results

Figure 7.7 UAV planning results

To sum up, Table 7.4 concludes the results of the four experiments. The error represents the deviation of the UAV actual path (red curve) compared to the ground-truth path (blue curve). The table shows that the error margin is small compared to the covered flying distance, which proves the high performance of the algorithms and their applicability in various navigation applications. Also shows that a slight fluctuation during landing as a constant error during all experiments.

Table 7.4 UAV planning quantitative analysis

Scenario	Waypoints	Distance [m]	Time [s]	Error [%]
1	21	14.89	38.30	3.30
2	43	39.53	72.93	4.03
3	37	32.52	64.29	3.25
4	22	16.88	41.29	5.50

7.5 iCab Communication Results

In this section, the results for the proposed schemes in Chapter 5, all experiments were carried using the iCab platforms and its on-board devices.

7.5.1 Setup

The testing environment was the off-road campus vicinity, at which the unstructured routes and the presence of unpredictable behavior of pedestrians increase the complexity. The campus includes several wireless networks access points, which are used for the WiFi experiments on the IEEE 802.15.4 standard for low-rate wireless personal area networks [228]. The WiFi heat-map of the campus is shown in Figure 7.8, which is mapped by performing a separate experiment through the whole campus. On the other hand, 4G tests took place in the same environment using a 4G-LTE router of 150 Mbps. The bandwidth of the wireless network is around 4 Mbps at downlink and 3.5 Mbps at uplink, while the 4G network has the speed around 2 Mbps in both directions. These values are estimated in ideal scenarios and they vary according to signal strength, channel load, and frequency among others.

7.5.2 Scenarios

In order to test the proposed schemes, three different scenarios were selected. Several experiments were performed for each scenario, using WiFi and 4G networks for the communication schemes. In Scenario I, both testing agents are static at different locations in the campus (Static-Static). While in Scenario II, one agent is static and the other is moving at different locations in the campus (Static-Dynamic). Finally, Scenario III, both testing agents are moving at different locations in the campus (Dynamic-Dynamic).

Accordingly, for the V2V communication tests, both vehicles were static in scenario I, one is static and one is dynamic in scenario II, and both are dynamic in scenario III. For the V2P and V2I communication tests, only the first two scenarios were tested, where the pedestrian or the infrastructure were always static and the vehicle was once static and another dynamic. This is due to the fact that all experiments evaluation were performed at the vehicle side. Figure 7.8 also shows the location of both vehicles and pedestrian testing zones, as noted by red oval shapes and blue cross respectively.

During each experiment, the following information is shared for every proposed communication scheme. In the V2V experiments, vehicle position, orientation, stats and tasks list are selected. For the V2P experiments, the vehicle position, orientation and alert notification

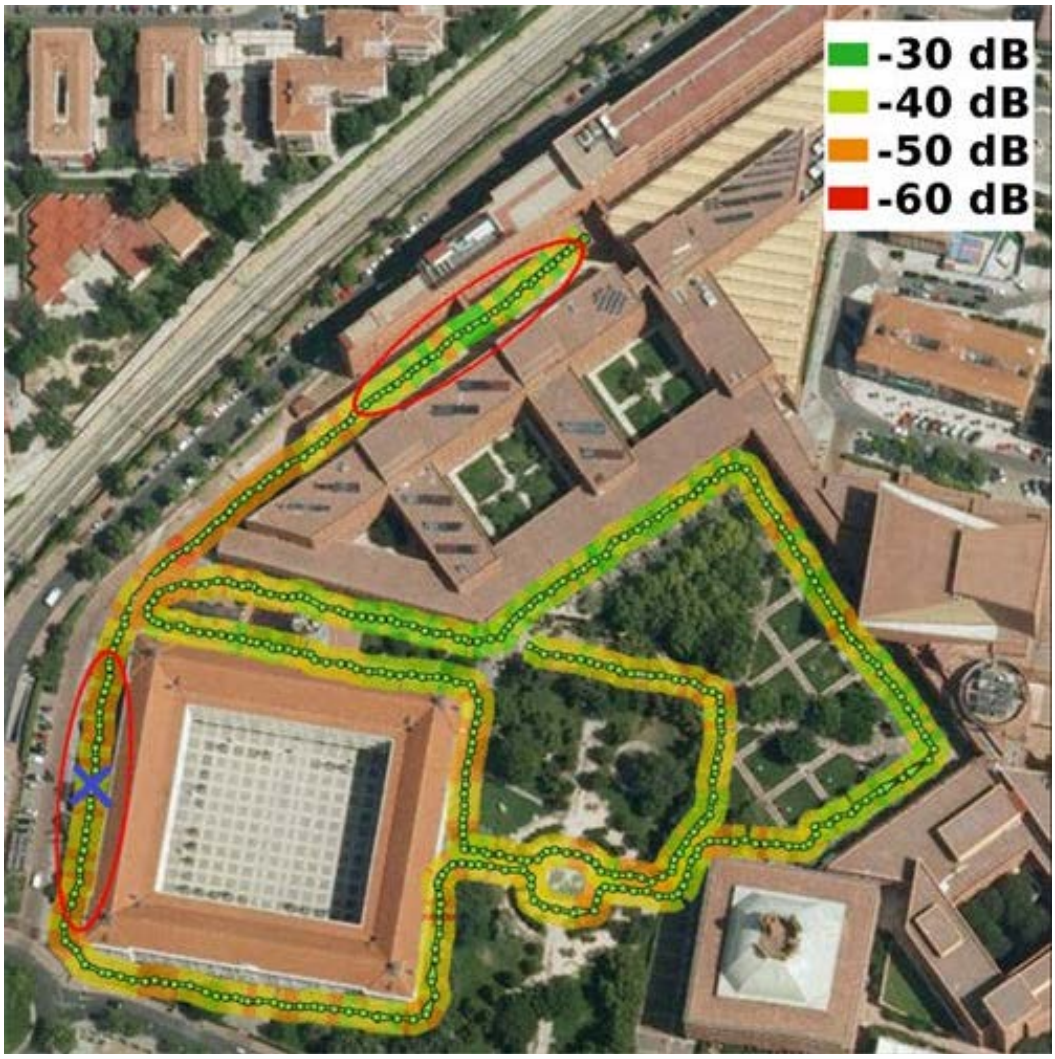


Figure 7.8 WiFi heat-map of the testing environment

messages are selected. Finally, for the V2I experiments, vehicle position, orientation, stats and requests information database are selected.

The selected scenarios require low bandwidth, thus all shared messages are transmitted in one single transaction. However, each of the proposed schemes is running at a different rate, for instance, inter-vehicle communication is at $20Hz$, communication with pedestrians is at $2Hz$ and communication with infrastructure is at $1Hz$.

7.5.3 Metrics

In order to prove the credibility and efficiency of the proposed communication schemes in the ITS field, several metrics are introduced to describe the characteristics of the system.

Since all the proposed schemes use TCP socket connections for the pairwise communication, the metrics aim to give information about the quality of these connections. From the many available indicators, the Round Trip Time (RTT) is selected. Due to the fact of its accuracy in measurement on the client side, also it evaluates the dynamism of the communication. The RTT, in this case, can be calculated by measuring the time between a data packet is sent and its acknowledgment arrives on the sender network interface. In all the scenarios the measurements were performed on the vehicle side, since the goal of the paper is to propose a muti-modal communication environment, from the point of view of autonomous vehicles in off-road scenarios. It is important to note, that measuring the round trip time on transport layer also takes the packet losses into account, by increasing the measured time. Thus, by analyzing the RTT, many information can be subtracted from it, such as:

- Bandwidth: since both the uplink and downlink bandwidth are used for sending the data and receiving the acknowledgment.
- Availability: by checking the outliers packets, it is inferred which ones were not delivered on time, which indicates the poor quality of the network.
- Smoothness: relying on the variance of the RTT, the smoothness of the network is evaluated, together with information about higher level parameters, such as jitter.

.

Based on the aforementioned, RTT is suitable to compare the results in every scheme individually. However, it can not be used directly to compare the different schemes to each other since RTT is highly dependent on the size of the transferred data, which varies from one scheme to another. The acceptable RTT value depends highly on the type of the communication. Since there are types, such as video or audio streaming, that are tolerant to higher delay but does not tolerate jitters, the deviation of the RTT value. And others, such as data transfer, can handle the variance of the response times, but really sensitive to the response time. It was concluded that $100ms$ as response time and $3-5ms$ as jitter guarantees smooth cooperation.

7.5.4 Qualitative and Quantitative Analysis

In this section results and a detailed description are given, in order to prove that the proposed system can be used to support autonomous agents. Box plots have been selected for data representation because it is possible to represent average, quartiles, variation, and outliers.

In all figures, along the horizontal axis, the results are divided according to the different scenarios and also grouped together based on the network type, WiFi or 4G. On the vertical

axis, the values of the RTT are measured on a logarithmic scale. The average is depicted with a horizontal line inside a box, which shows the quartiles of the dataset. The whiskers extend to show the rest of the distribution, except for points that are determined to be outliers.

A value is considered to be an outlier if it fulfills one of the following criteria:

$$\begin{aligned} outlier &\leq Q1 - 1.5 * IQR \\ outlier &\leq Q3 + 1.5 * IQR \end{aligned} \tag{7.3}$$

where the $Q1$ or $Q3$ determines the quartile and the IQR is the interquartile range. These outliers are also highlighted with small filled rhombuses in results figures.

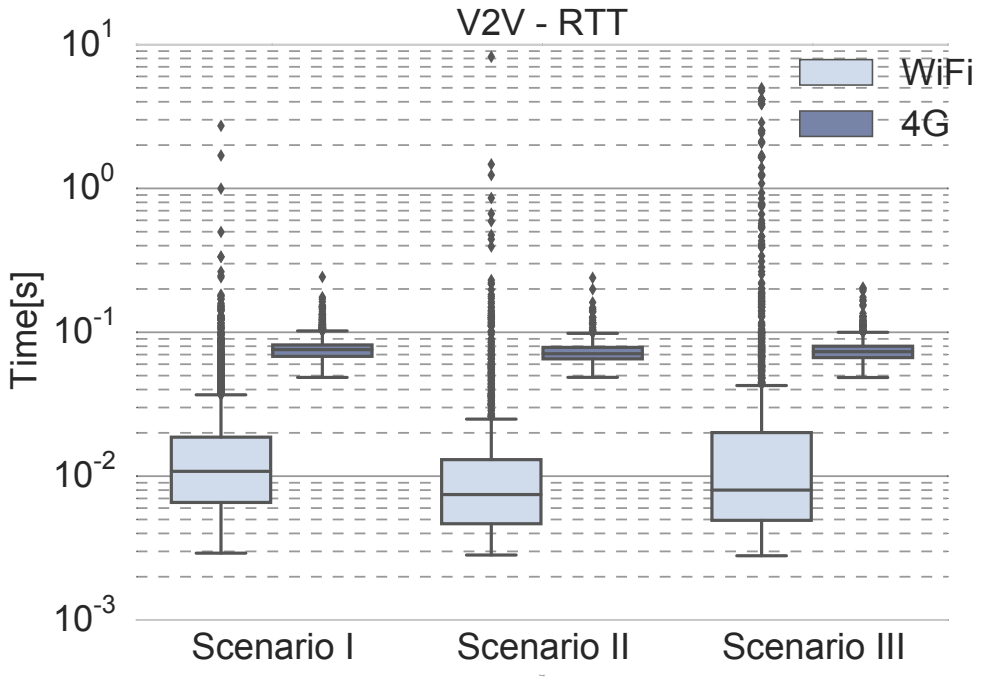


Figure 7.9 Results of the V2V communication

As it is depicted in Figure 7.9, in the case of V2V, the communication over WiFi is faster than 4G in most of the scenarios, according to the average values and on the quartiles. This is absolutely straightforward since in the WiFi case both of the vehicles are connected to the same subnetwork. However, in the 4G case, the data has to be conveyed to the operator's router via the base station, and then backward as well, which results in higher response time. The variances of the measured values are almost the same in both network configurations and all scenarios, as shown in Table 7.5. The difference appears when the outliers are taken

into account, where the WiFi backed solution has higher number and values for the outliers, in comparison to the 4G. As detailed in Table 7.5, during scenario II there were 8s with no communication, furthermore, higher delays can be found in scenarios I and III. However, when using 4G connection, the outliers remain under 0.24s. This due to the availability difficulties of the campus WiFi network, as shown in Figure 7.8.

Table 7.5 Main results of the V2V communication

Type	**Mean[ms]**			**IQR[ms]**			**Max[s]**		
Scenario	I	II	III	I	II	III	I	II	III
WiFi	11	8	8	12	8	15	2.72	8.21	4.98
4G	75	72	73	14	13	13	0.24	0.24	0.20

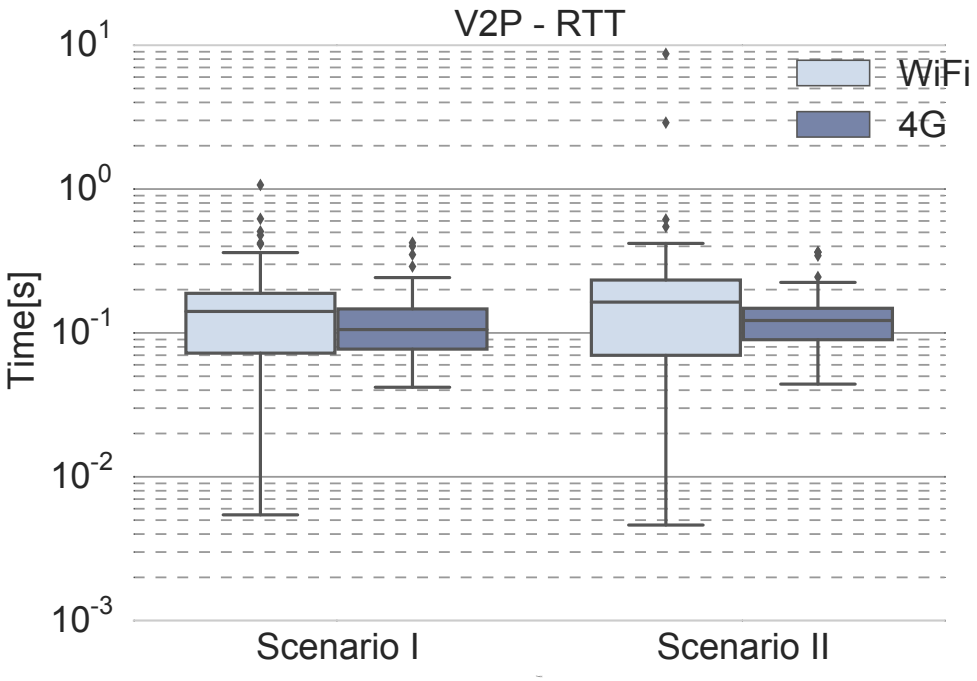


Figure 7.10 Results of the V2P communication

Regarding the V2P communication, the average of the RTT values and the IQR are similar between WiFi and 4G. These values are expected since the pedestrian's device and the vehicle is not connected to the same subnetwork. Also, based on the variance, it can be interpreted that the WiFi connection is faster, due to the higher bandwidth compared to 4G. Concerning the outliers, the results were different in each scenario. For scenario I, the

WiFi and 4G connection show almost the same behavior, while in scenario II, WiFi has significantly higher delays for the same reasons described in V2V results. Thus in the case of V2P communication, the proposed VPN over 4G architecture, does not only have the advantage of the persistent connection but also have the same performance as WiFi.

Table 7.6 Main results of the V2P communication

Type	**Mean[ms]**		**IQR[ms]**		**Max[s]**	
Scenario	I	II	I	II	I	II
WiFi	140.5	164.4	116.1	163.2	1.06	8.70
4G	105.3	123.3	69.6	58.6	0.41	0.36

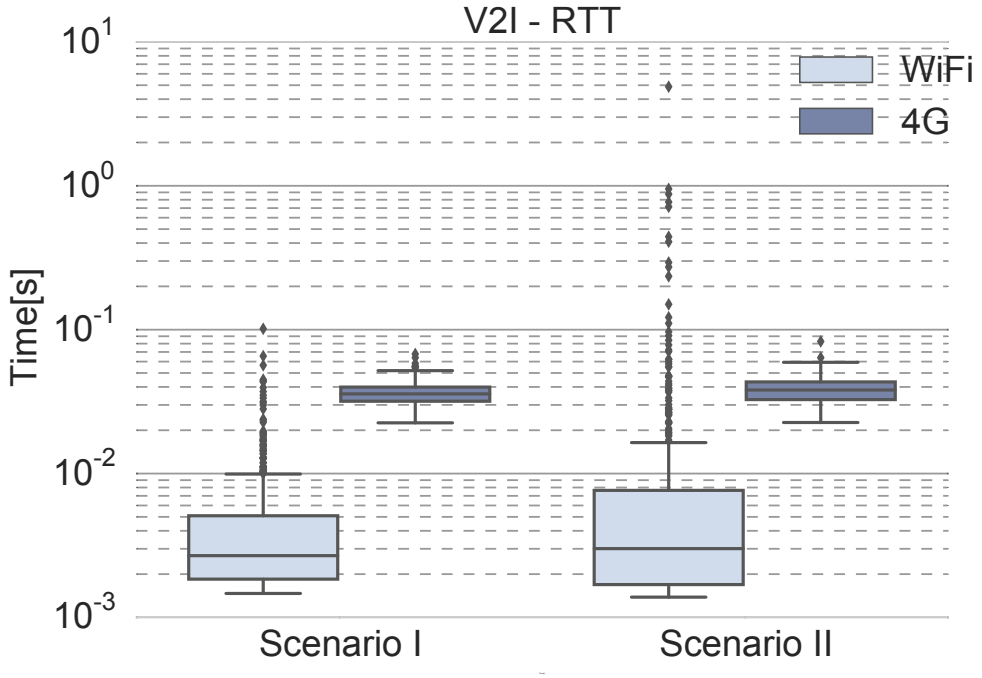


Figure 7.11 Results of the V2I communication

Finally, the results of the V2I communication are depicted in Figure 7.10 and also shown by Table 7.7. Opposite to the other schemes, the V2I communication does not use VPN, instead, the entities communicate directly through the internet. The average and the quartiles values in the WiFi configuration were lower than the ones in 4G connection, due to the fact that both, the vehicle and the infrastructure, were connected to the same network. The smoothness of the network according to the measured variance in the WiFi and the 4G

connections are nearly the same overall experiments in both scenarios. As expected, there was a higher amount of outliers than previous schemes, even higher in the WiFi connection than in the 4G, as seen in previous studies. The importance of having a stable connection is keeping the status of the vehicles up to date and also deliver the request instantaneously.

Table 7.7 Main results of the V2I communication

Type	**Mean[ms]**		**IQR[ms]**		**Max[s]**	
Scenario	I	II	I	II	I	II
WiFi	2.7	2.9	3.3	5.9	0.10	4.88
4G	35.1	37.01	7.8	10.6	0.07	0.08

7.6 iCab Localization Results

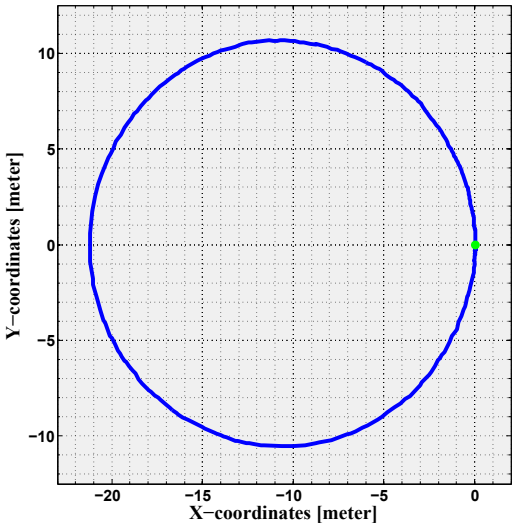
In this section, the results for the proposed localization and covariance estimation approaches in Chapter 4, all experiments were carried using the iCab platforms and its on-board devices.

7.6.1 Scenarios

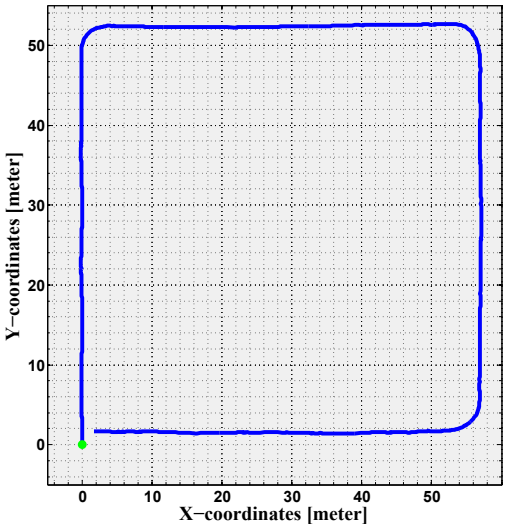
In order to validate the proposed approach, several scenarios are designed and tested over various experiments. This section describes the used platform for the real-word experiments, the testing environment, the designed scenarios, and the selected evaluation metrics. The testing environment was the off-road vicinity of the campus, which has free pedestrian areas and surrounded with many buildings. In this environment, three scenarios were designed to evaluate the proposed approach, and each scenario was experimented three times under different conditions. All scenarios are depicted in Figure 7.12.

Scenario I

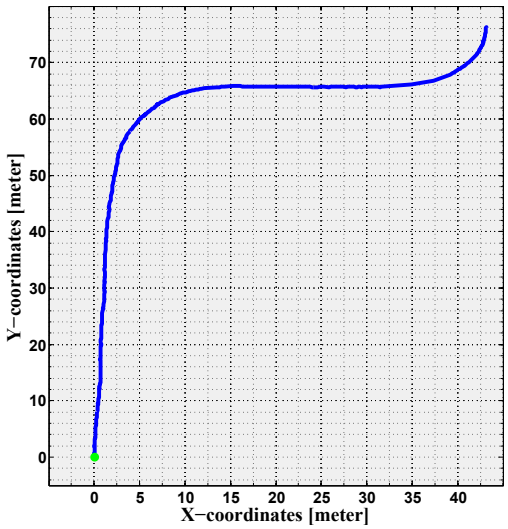
The first scenario was designed as a circle of total diameter of $22m$, in which the iCab steering wheel was adjusted to 8.5^o and average velocity of $5km/h$. In this case, the theoretical path was designed as a pure circle with the same diameter to be compared with the obtained odometries. The scenario was selected to evaluate the proposed approach performance in optimizing the localization in simple circular motion. Moreover, it is a closed-loop, thus the vehicle end point is the same as the starting point.



(a) Scenario I



(b) Scenario II



(c) Scenario III

Figure 7.12 Visual demonstration of the selected scenarios

Scenario II

The second scenario was designed as a quadratic shape of total length of $54m$ and width of $56m$. This was also a closed shape, where the theoretical path was designed as a right-angled quadratic shape. The iCab followed the shape with average velocity of $5km/h$ and rotating with sharp 30^o around the corners. The scenario was selected to evaluate the proposed approach performance in optimizing the localization in both straight-line and curved motions.

Scenario III

The third and last scenario was designed as a trajectory from one building to another in the campus. The selected points are part of the pick-up / drop-off points of the iCab project. The theoretical path was obtained from the path planner to be compared with the obtained odometries. The path consists of multiple curves, straight-lines, dynamic obstacles, and it is one of the normal trajectories that the iCab follows in its daily operation. The iCab followed the path with average velocity of $5km/h$ and a maximum of 15^o steering angle during curvatures.

7.6.2 Evaluation Metrics

In order to evaluate the true potential of the proposed algorithm, the performance and results are compared relative to different values of constant covariances in a fusion algorithm, and the *robot_localization* package was selected for the fusion [205]. The evaluation metrics for the Ackermann model are calculated for both, translation and orientation. Firstly, the mean and maximum error percentages of the translation; which are calculated as shown in Equations (7.1) and (7.2) respectively.

As for the orientation, the mean and maximum error in the orientation are divided by the total distance covered by the vehicle, which are calculated as shown in Equations (7.4) and (7.5) respectively.

$$OE_{mean}[^o/m] = \frac{\frac{1}{N}\sum_{k=1}^{N} \hat{\theta}_k - \theta_k}{TotalDistance} \tag{7.4}$$

$$OE_{max}[^o/m] = \frac{\frac{1}{N} max(\hat{\theta}_k - \theta_k)}{TotalDistance} \tag{7.5}$$

where $\hat{\theta}_k$ is the estimated orientation of the vehicle and θ_k is the true orientation of the vehicle at time step k.

7.6.3 Qualitative and Quantitative Analysis

In the experiments, five different odometries were executed in the iCab platform; the lidar as the reference odometry, GPS as the exteroceptive sensor odometry and wheel encoders as the propriocetpive sensor odometry. Additionally, visual odometry and compass orientation were included as more inputs to the fusion algorithm. The covariance estimation was estimated for the propriocetpive sensor based on the available covariance of the exteroceptive sensor. In

order to show the efficacy of the covariance estimation algorithm, the results are compared to the results using different values of constant covariances. The values used are percentages of the True Variances (TV) of both the steering encoder and the translation encoder, which are calculated retroactively from the data of the experiments. The parameters values used for the covariance estimation algorithm is the same in all scenarios, to ensure that the algorithm does not need different parameters for different scenarios. During all experiments, the exteroceptive sensor parameters were as follows: α was set to 0.5, c_e and c_s were set to 0.05, and ε_e and ε_s were set to 0. Finally, the constant values of covariances are selected depending on the error values for each experiment. As shown in the tables below, 125% of the TV gave higher error than 25% of the TV. Observing these results, smaller percentages of the TV were used in order to reach better results. And lower than 2.5% of the TV were omitted, because they gave larger errors.

Scenario I

Table 7.8 shows the experimental quantitative results of the first scenario, comparing the proposed approach of covariance estimation against the TV at four different values. The obtained results show the covariance estimation algorithm outperforming all TV values in both translation error and orientation error. All performed trials of TV obtained near results at different values. There might be a chance that with more trials of different values of TV, one of them could outperform the proposed approach, however, the tuning issue consumes time and efforts.

Table 7.8 Mean of the 3 scenario I experiments results

Metrics	**Mean [%]**		**Mean [$^o/m$]**	
	TE_{mean}	TE_{max}	OE_{mean}	OE_{max}
Adaptive	1.743	3.043	0.0029	0.097
True Variance	6.756	13.853	0.249	0.612
2.5% TV	1.796	3.598	0.077	0.227
5% TV	2.619	5.705	0.099	0.302
25% TV	4.061	7.746	0.164	0.349
125% TV	7.613	14.772	0.263	0.667

Scenario II

Table 7.9 shows the experimental quantitative results of the second scenario, comparing the proposed approach of covariance estimation against the TV at four different values. The

obtained results show that the proposed approach was able to outperform the TV in the translation errors. However, it obtained minimal error of 0.001^o in the mean orientation in comparison to one of the TV, despite having the best maximum orientation error.

Table 7.9 Mean of the 3 scenario II experiments results

Metrics	**Mean** [%]		**Mean** [$^o/m$]	
	TE_{mean}	TE_{max}	OE_{mean}	OE_{max}
Adaptive	3.047	4.717	0.020	0.089
True Variance	13.034	44.662	0.019	0.109
2.5% TV	4.942	8.259	0.019	0.111
5% TV	6.498	11.451	0.019	0.103
25% TV	9.633	21.433	0.020	0.111
125% TV	13.593	60.060	0.019	0.110

Scenario III

Table 7.10 shows the experimental quantitative results of the second scenario, comparing the proposed approach of covariance estimation against the TV at four different values. The obtained results show that the proposed approach was able to outperform the TV maximum errors in both translation and orientation. However, for the mean errors, it obtained a minimal error value of 0.098% in the mean translation error, and a minimal error value of 0.003^o in the mean orientation error. This is taking into consideration that the best mean translation error was obtained at 2.5% of the TV, while the best mean orientation error was obtained at the exact value of the TV, which implies that the proposed approach outperforms in general all TV values.

Table 7.10 Mean of the 3 scenario III experiments results

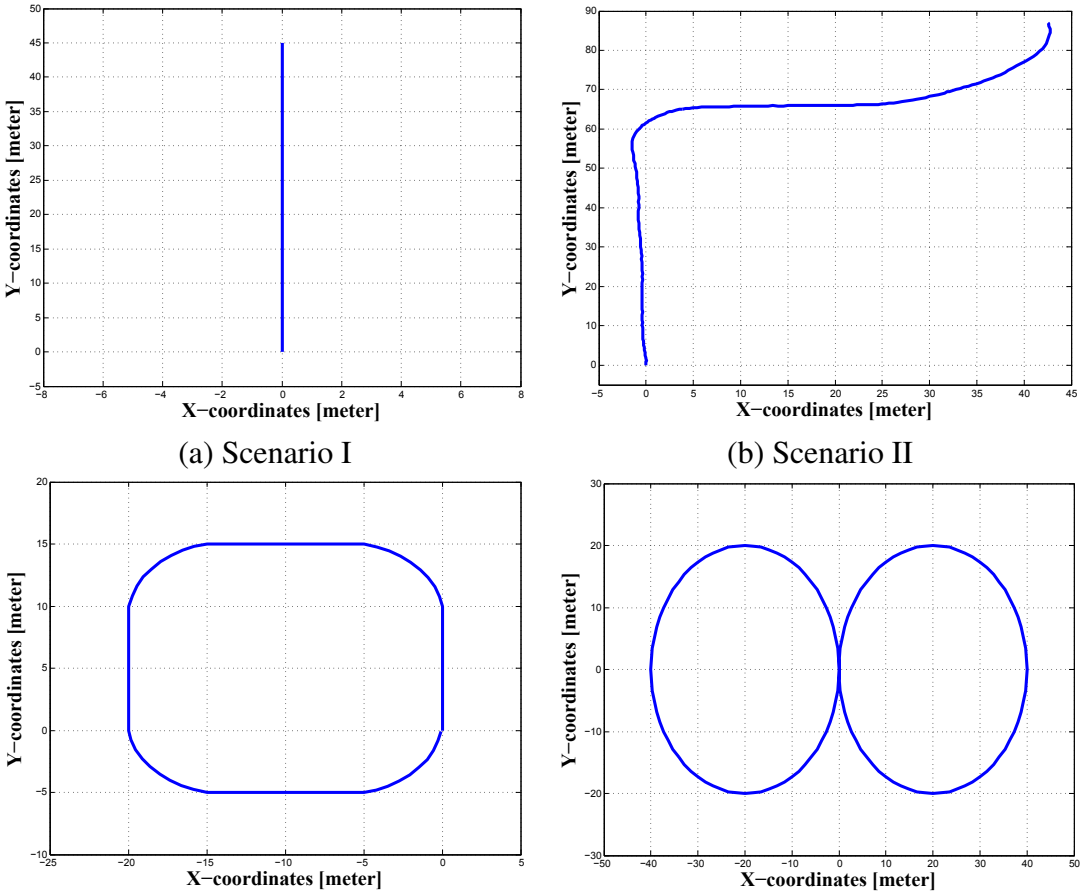
Metrics	**Mean** [%]		**Mean** [$^o/m$]	
	TE_{mean}	TE_{max}	OE_{mean}	OE_{max}
Adaptive	4.473	7.545	0.025	0.085
True Variance	15.841	56.230	0.022	0.167
2.5% TV	4.375	8.971	0.023	0.090
5% TV	4.613	10.372	0.024	0.120
25% TV	7.627	21.715	0.025	0.147
125% TV	13.508	46.674	0.024	0.155

7.7 iCab Planning Results

In this section, the results for the proposed planning approach in Chapter 4 are presented, all experiments were carried using the iCab platforms and its on-board devices.

7.7.1 Setup

The experiments are divided into two parts, the simulation environment and the real-world environment. On the one hand, the simulator platform used to test all the algorithms is the ROS Gazebo simulator. The *car_demo* simulator package was used [143]. The simulation on a different model than the actual experimental platform was essential to the validation of the controller, because it demonstrated the flexibility of the controller and the simplicity of tuning it to different vehicle models. On the other hand, the proposed iCab platform was used for the test in the real scenarios.



(a) Scenario I

(b) Scenario II

(c) Scenario III

(d) Scenario IV

Figure 7.13 Visual demonstration of the selected scenarios

7.7.2 Scenarios

Four different scenarios were defined using ROS path message, which contains an array of X and Y coordinates, in addition the heading angle at each waypoint. Each scenario path planning points was saved and used as input for all path tracking approaches for comparison.

Scenario I

The straight line path is the simplest test that can be performed, which is shown in Figure 7.13 (a). The main maneuver for the vehicle is to demonstrate the tracking capability on a straight path, the length of this path is $45m$. Experiments on this path were performed at velocities of $1m/s$, $2m/s$ and $3m/s$.

Scenario II

The normal path captures a variety of driving scenarios and a good simulation to real-world driving, which is shown in Figure 7.13 (b). This path is important to demonstrate general qualitative performance and path tracking capabilities. Experiments on this path were performed at velocities of $1m/s$ and $2m/s$.

Scenario III

The square shaped path of $20m$ side-length, shown in Figure 7.13 (c), is essential in validating a path tracking algorithm, to test the capability of the algorithm in keeping track of the path and the robustness of the controller. The closed loop path validates the tracking approaches when handling accumulated error. Experiments on this path were performed at velocities of $1m/s$ and $2m/s$.

Scenario IV

This path does not reflect a normal driving scenario, however it was selected since it provides valuable insight into the handling of a vehicle, path is shown in Figure 7.13 (d). This experiment was tested in simulator only, since in real-world environment had free area size limitation. Experiments on this course were performed at velocities of $1m/s$, $2m/s$ and $3m/s$.

7.7.3 Metrics

In order to evaluate the proposed approach efficiency and compare it to the other approaches, the vehicle localization was compared to the theoretical path planning input. Therefore, four metrics were selected to evaluate the data in quantitative and qualitative manners.

Relative Position Error (RPE)

It shows how far the vehicle is from the path as the lateral deviation. The value is calculated as a percentage, where the Euclidean distance of each position is calculated with respect to the path and then the mean of these values is divided by the total path length.

Maximum Position Error (MPE)

It shows the maximum instant where the vehicle was deviated from the path. The value is in meters, which is calculated as the maximum Euclidean distance error from all recorded positions.

Relative Rotation Error (RRE)

It shows how accurate the vehicle was oriented with respect to the path planning input. The value is calculated as degrees per meter, which by subtracting the vehicle heading angle from the waypoint required heading angle at each position over the path then the mean of these values is divided by the total path length.

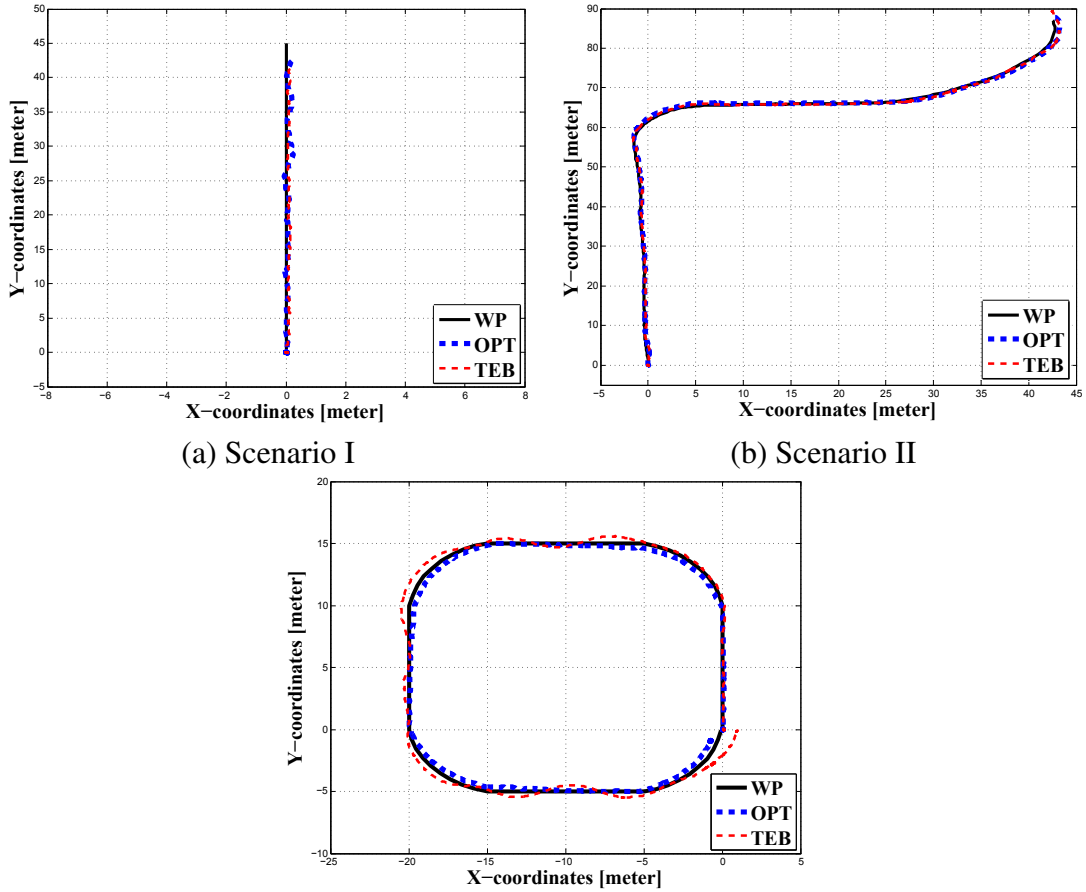
Battery Consumption (BC)

It shows the the battery consumption by the vehicle during the path tracking. The value is calculated as the energy in kW, which is drawn from the batteries, taking into consideration all mounted devices consumption, and measuring power withdrawn by the translation motor at every navigation command. Therefore, the control cost is compared for each tracking method.

7.7.4 Qualitative and Quantitative Analysis

The experimental work resulted in collecting data from a total of 72 real world experiments and evaluating four metrics from each. In this section, two different path tracking approaches were tested, which are: the proposed optimal path tracking algorithm (OPT), *teb_local_planner* ROS package (TEB) [208]. The most relevant results are summarized in

the following subsections. Both path trackers were fully tuned and optimized to run on the test platform to ensure a fair comparison.



(a) Scenario I

(b) Scenario II

(c) Scenario III

Figure 7.14 Visual demonstration of the selected scenarios

Scenario I

The experiment results in this scenario are shown in Figure 7.14 (a) for the $1m/s$ velocity experiment. The OPT showed better stability in comparison to the other approach. As shown in Table 7.11, the performance of OPT and TEB were relatively close in the results, though the TEB performance failed at higher velocities. In conclusion, OPT outperformed the other approach in the RPE, RRE and BC, however the TEB obtained less MPE.

Table 7.11 Scenario I - Experiments Results

Metrics	OPT	Teb
RPE [%]	0.3009	0.3928
MPE [m]	0.5224	0.5101
RRE [^{o}/m]	1.7297	1.8548
BC [kW]	6.8953	10.4321

Scenario II

Figure 7.14 (b) shows the performance of the path tracking approaches in following a predetermined path at $1m/s$ velocity. The trackers were able to follow the path accurately, though the OPT has more aggressive response. As shown in Table 7.12, the OPT outperformed the other approaches in the RPE and BC, however TEB obtained less values in MPE and RPE.

Table 7.12 Scenario II - Experiments Results

Metrics	OPT	Teb
RPE [%]	0.2284	0.2372
MPE [m]	0.9089	0.6595
RRE [^{o}/m]	3.4900	3.0477
BC [kW]	25.7941	39.4247

Scenario III

Figure 7.14 (c) shows the performance of both OPT and TEB at $1m/s$ velocity, at which they were able to track the path with minimal errors. Table 7.13 summarizes the quantitative results, where OPT outperformed TEB in the RPE and RRE, and TEB outperformed in the MPE.

Table 7.13 Scenario III - Experiments Results

Metrics	OPT	Teb
RPE [%]	0.3966	0.6282
MPE [m]	1.1621	1.0548
RRE [^{o}/m]	4.1319	10.1265
BC [kW]	17.8671	24.7688

Simulation Results

The four scenarios were tested successfully in the simulation environment and results are summarized in Table 7.14 as the average of all carried-out experiments at different velocities profiles. Simulation experiments offers a deeper and more comprehensive study between the performance of the approaches. Following the same evaluation metrics, excluding BC, this tool allowed a larger number of experiments with different velocity profiles. The comparison shows relatively close errors for both approaches.

Table 7.14 Simulation Experiments Results

Simulator Results					
Scenario I			***Scenario II***		
Metrics	**OPT**	**TEB**	**Metrics**	**OPT**	**TEB**
RPE [%]	0.5966	0.5770	**RPE %**	0.1689	0.1341
MPE [m]	0.5319	0.5357	**MPE [m]**	0.4179	0.4073
RRE [^{o}/m]	0.7325	0.8896	**RRE [^{o}/m]**	2.6592	2.1174
Scenario III			***Scenario IV***		
Metrics	**OPT**	**TEB**	**Metrics**	**OPT**	**TEB**
RPE [%]	0.2214	0.2217	**RPE [%]**	0.1771	0.1787
MPE [m]	0.5082	0.4986	**MPE [m]**	0.9019	0.8722
RRE [^{o}/m]	1.5386	1.5964	**RRE [^{o}/m]**	1.0194	0.6939

7.8 iCab Platooning Results

In this section, the results for the proposed platooning approach in Chapter 5, all experiments were carried using the iCab platforms and its on-board devices.

7.8.1 Scenario Description

The selected real-world scenario covers all testing parameters. The process is as follows, the system receives transport request, from one point to another. This request is for 4 passengers, since each vehicle has a maximum capacity of 2 passengers only, the two vehicles have to carry out this task. Accordingly, in order to achieve an optimal cooperative driving, one vehicle should be responsible for the main environment perception, path planning and autonomous navigation from the starting point to the goal point, while the other vehicle should communicate with the first vehicle and follow it to the destination point. In other

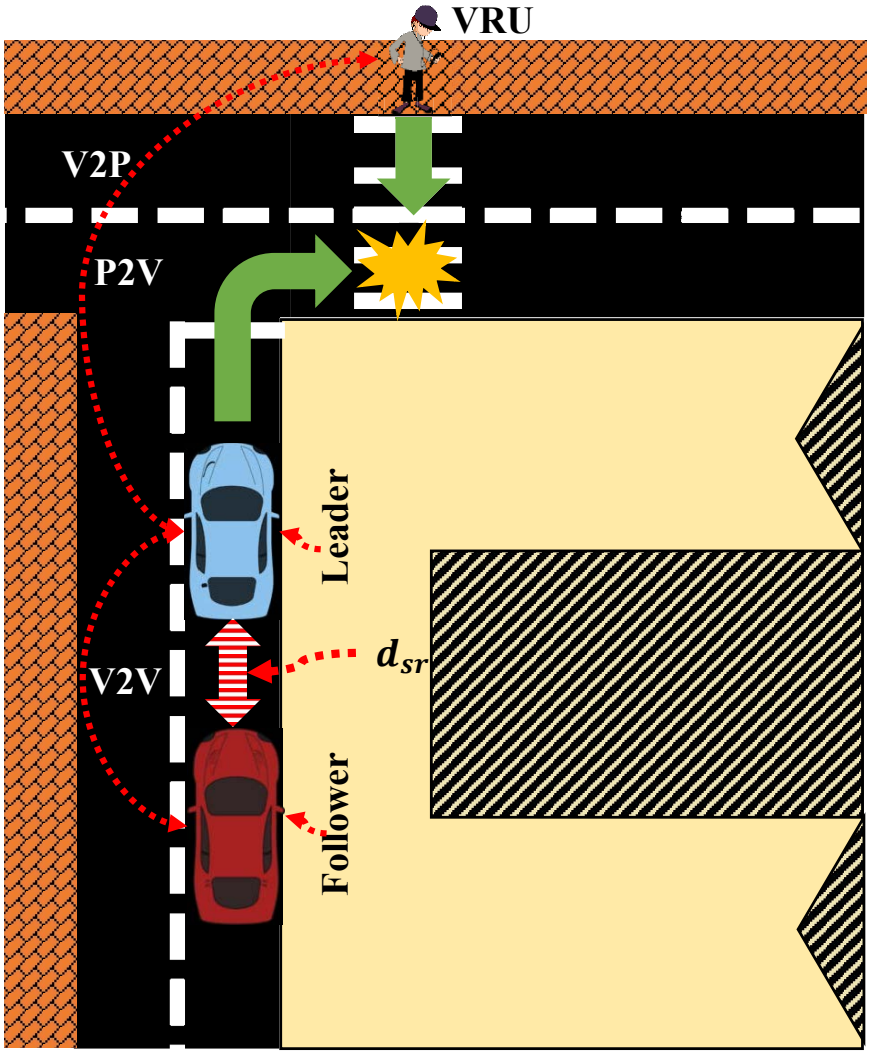


Figure 7.15 Two autonomous vehicles platooning scenario with VRU crossing their designated path

Meanwhile the vehicles are navigating to the destination point, a distracted VRU is trying to cross their designated path of the vehicles. This plot is directed to simulate a real life scenario in a smart city, where the distracted user is crossing the road without paying attention to the road, and the autonomous vehicles are following each other to a collision point with the user without any mean of detecting the user with the on-board perception sensors.

The scenario studies the efficiency of V2V communication and the platooning algorithm in cooperative autonomous driving. Moreover, it reviews the advantages of V2P and P2V communications in detecting the pedestrian in advance and sending a warning message to the pedestrian and a notification to the vehicle.

7.8.2 Qualitative and Quantitative Analysis

Real life experiments were performed inside the campus vicinity as an off-road environment. Both iCab platforms were used in these experiments as the autonomous vehicles to test the cooperative driving approach. And several volunteers participated in the experiments as the VRU. The scenario was tested several times to obtain enough results for analysis. All recorded data took place in the ROS architecture with time-stamp.

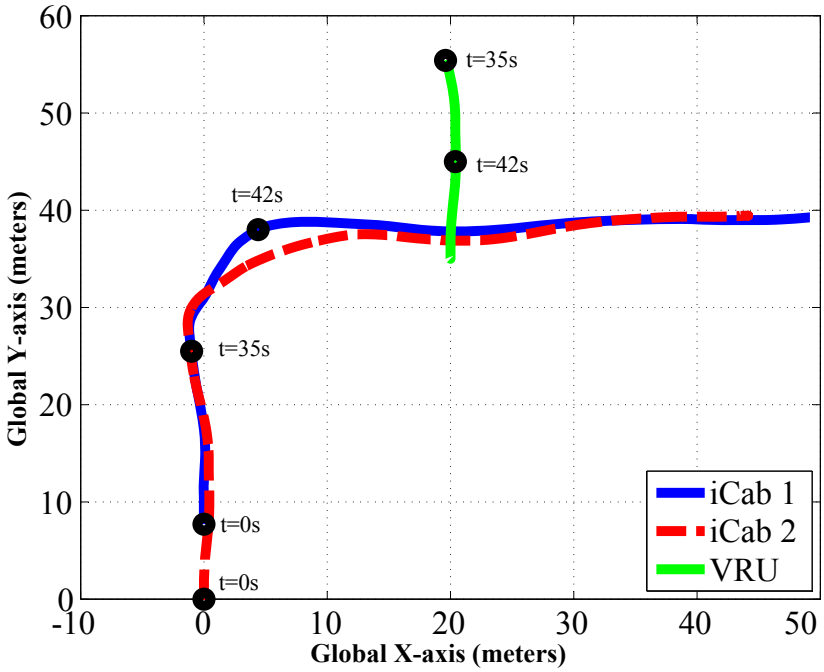


Figure 7.16 Trajectories of iCab 1 (Blue), iCab 2 (Red), and VRU (Green)

Figure 7.16 shows the navigated trajectories of the leader vehicle (iCab 1) and follower vehicle (iCab 2) on XY plane, along with the trajectory of the pedestrian (VRU). The leader vehicle was set to drive at speed of 5 km/h and the follower vehicle adjusted its speed and the spacing distance accordingly. The figure also shows that the follower vehicle is moving along the leader vehicle trail, while maintaining the desired distance with minimal error.

Further details about the spacing distance error values between the follower and the leader vehicles, in both X and Y directions relative to the vehicle, are presented in Figure 7.17 (a) and (b) respectively. It is seen from the figure that the tracking errors in the lateral direction (vehicle X-axis) osculates around the reference and settle to 0.65 meters. However, for the tracking errors in the displacement longitudinal direction (vehicle Y-axis), it decays to almost zero meters with a maximum peek of 1.3 meters error during the turning part of the scenario.

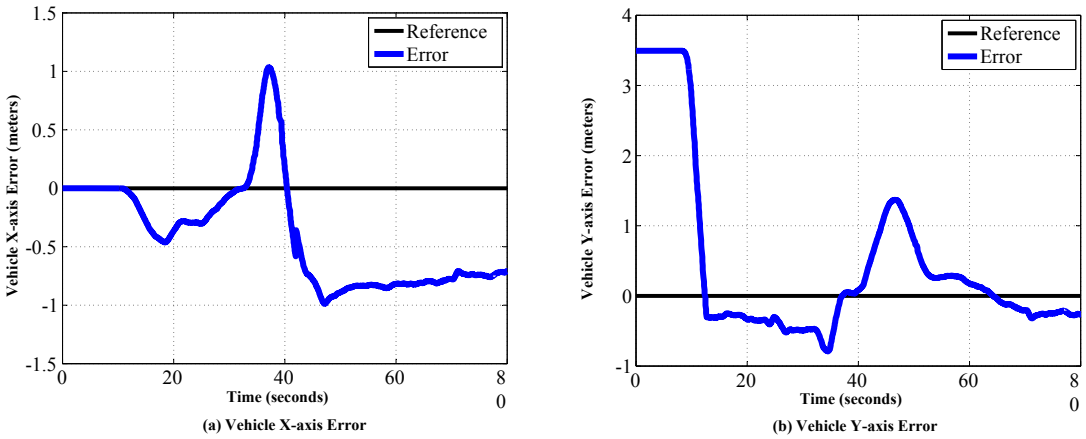


Figure 7.17 Spacing distance error between follower and leader vehicles

Figure 7.18 displays the results of the leader vehicle obstacle detection algorithms against time, in addition to the moment of which the warning notification message was sent through the P2V communication. Since the pedestrian is crossing on an orthogonal direction, there is no possible way for the sensors of detecting him. It is considered as a blind spot due to an obstruction. Hence, the advantage of the developed system is the anticipation of VRU crossing the vehicles path approx. 7 seconds in advance. The leader vehicle then uses V2P communication to warn the VRU, and V2V communication to inform the follower vehicle of the presence of the VRU.

The qualitative results prove the viability of the proposed approach for cooperative autonomous driving, in addition to the advantages of the V2X communications in enhancing the environment perception system.

Tables 7.15 and 7.16 present the quantitative analysis for the overall system and compare the results with the use of V2X communication and without them. The results are obtained from performing the experiments on the same scenario and under the same condition, once with the communication schemes active and the other with the communication schemes inactive, then calculate the average value of the error over the whole path.

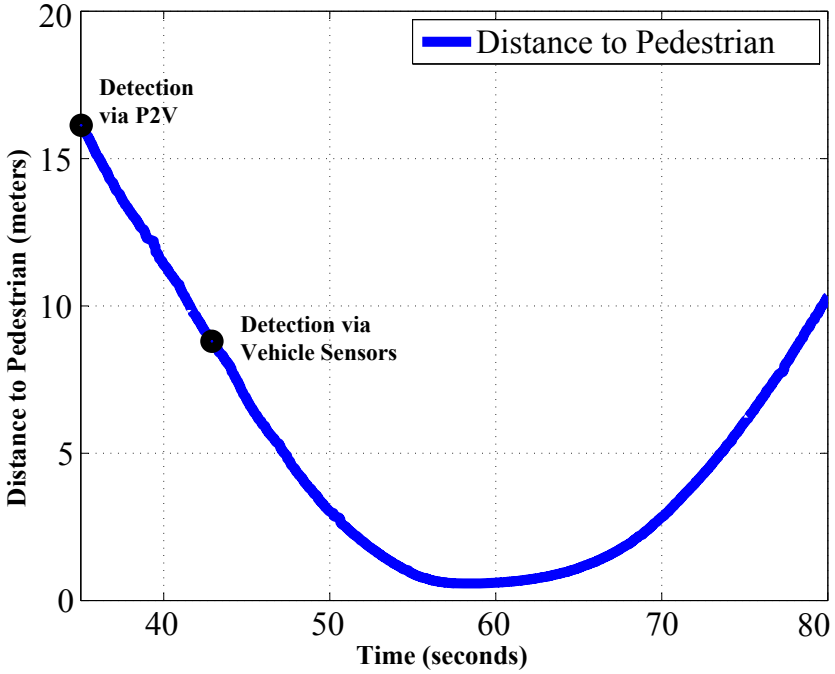


Figure 7.18 Euclidean distance between the pedestrian and the vehicle

In Table 7.15, VRU detection statistic are calculated based on the Euclidean distance between the vehicle and the pedestrian at the moment of detection. The leader vehicle was able to detect the VRU in both cases, however with the communication inactive, the detection was late which made the decision of braking be more aggressive. On the other hand, for the follower vehicle, it was not able to know that there is a pedestrian in the route with the communication inactive.

Table 7.15 VRU detection statistics [meters]

	With V2X	Without V2X
Leader Vehicle	16.13	8.79
Follower Vehicle	21.32	N/A

In Table 7.16, the leader vehicle errors are estimated based on the deviation from the route obtained by the path planning algorithm in the sequencer layer, therefore the values are the same in both cases. However, for the follower vehicle errors, the values are estimated based on the deviation from the trail of path made by the leader vehicle. It is shown that the error almost is doubled in the case of no communication.

Table 7.16 Tracking error statistics [meters]

	With V2X	Without V2X
Leader Vehicle	0.89	0.89
Follower Vehicle	0.43	0.93

7.9 MRTA Results

In this section, the results for the proposed task allocation approach in Chapter 6. Real world experiments were carried using the iCab platforms and its on-board devices.

7.9.1 Selected Scenarios

In order to test the proposed approach, several scenarios are selected in both simulation and real-world. The simulation scenarios are selected from well-known benchmarks of mTSP, this is in order to have the optimal cost available for comparison. Each scenario consists of a different number of cities, which are distributed over the environment in various locations. Figure 7.19 shows the four selected scenarios, where the top left one is Christofides/Eilon with 51 cities, top right is Berlin (Groetschel) with 52 locations, bottom left is Christofides-Eilon with 76 cities and bottom right is Rattled Grid (Pulleyblank) with 99 cities [229, 230]. Each scenario has only one depot, which is represented by the red marker in the graphs.

On the other hand, the real-world scenario was designed in a way to evaluate the functionality of the proposed solution and the architecture in the platforms. The scenario involved three users using the application to create transportation requests. The three users had different starting points and different destinations, moreover the request time was close to each other to evaluate how the vehicles are going to respond. Figure 7.20 shows the environment map with the vehicles and passengers locations, the two vehicles are represented with the golf-cart clip-art, the passengers representation are marked in three different colors, a circular shape of the same color for the desired destination. The experiment was video recorded and its results discussion is in the next section.

7.9.2 Evaluation Metrics

The proposed hybrid optimization-based solution is used to solve each scenario of the MRTA problem and the results are recorded for evaluation. In order to evaluate the quality, two evaluation metrics are introduced, which are the allocation cost and computational time. The

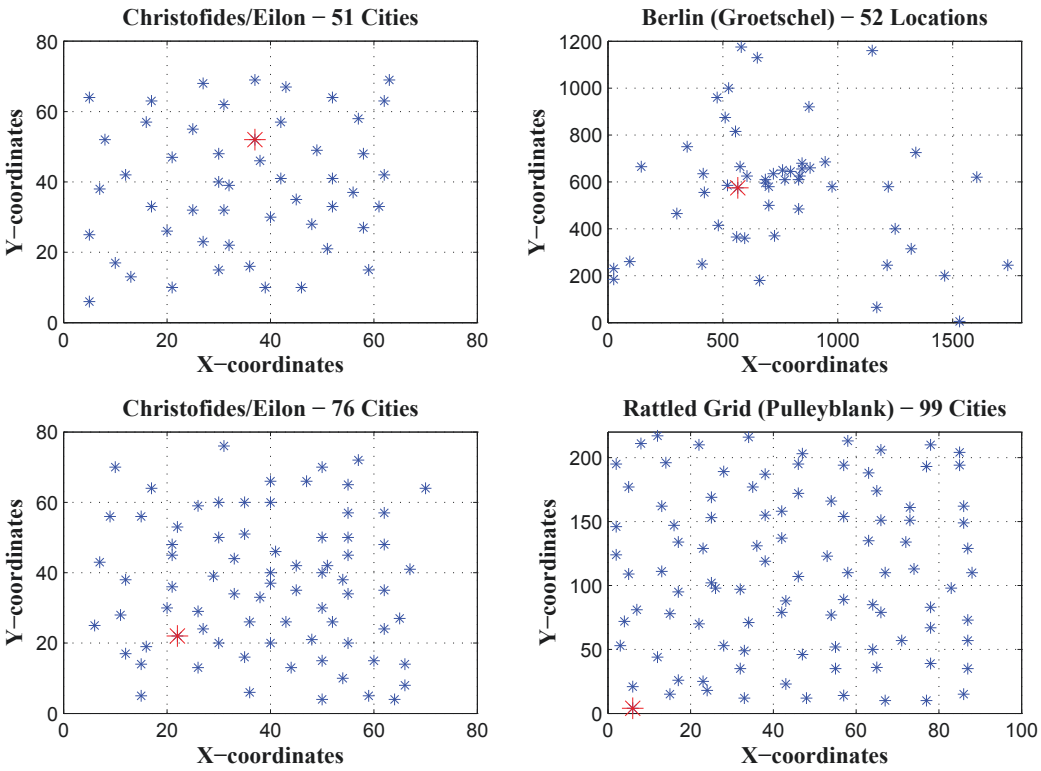


Figure 7.19 mTSP selected benchmark scenarios, red marker is the depot

first evaluation metric is the allocation cost of the best allocation found. The allocation cost is calculated based on the objective function. Thus two allocations costs are computed, one is the MinMax cost, which represents the length of the longest sub-tour in the allocation, and two is the total overall cost of all sub-tours. The second metric is the computational time required by the algorithm to reach the best solution. The timer starts after the databases of the vehicles and requests are read and stop when the algorithm stops, then the elapsed time is reported. Since the computational time calculation is depending on the machine, the computer used for all experiments has the specifications in Table 7.17.

Table 7.17 System Specifications

Processor	4 Cores, 4 Threads @3.8GHz
Memory	16GB DDR4-2133

7.9.3 Comparative Study

In this section, a comparative study is conducted between the proposed approach results and the reference optimal results presented in [230]. The optimal results are obtained for the

Figure 7.20 Environment map with the real-world scenario vehicles and passengers locations

four well-known benchmarks, which are described in the scenarios subsection. The authors in [229] adjusted the benchmark settings, to have all vehicles located at the depot and it is required to visit all locations in the scenario, such that each location is visited exactly once, then return to the depot.

Table 7.18 summarizes all the results from both the optimal solutions costs, compared to the obtained solutions costs, in addition to the computational time for the obtained solutions costs. The optimal costs are obtained using IBM CPLEX, it took 96 hrs for the $eil-51$ dataset, 120 hrs for the $berlin-52$ dataset, 168 hrs for the $eil-76$ dataset, and 216 hrs for the $rat-99$ dataset. On the other hand, the reported values of the obtained costs are the mean of the 25 experiments of each scenario.

Table 7.18 Benchmarks comparative results

Benchmark	Vehicles	Optimal MinMax	Optimal Total	Obtained MinMax	Obtained Total	Comp. Time (sec)
eil-51	2	222.73	444.33	236.49	470.97	124.86
	3	159.57	477.15	167.11	498.64	137.06
	5	123.96	615.19	129.36	629.64	158.86
	7	112.07	762.83	116.87	764.24	182.67
berlin-52	2	4110.21	8217.94	4315.83	8630.19	131.94
	3	3244.37	9591.15	3387.35	10115.18	140.97
	5	2441.39	12084.90	2509.84	**11969.61**	163.10
	7	2440.92	16768.79	2491.83	**15628.95**	194.21
eil-76	2	280.85	561.48	307.68	613.75	281.89
	3	197.34	587.65	210.97	632.07	292.83
	5	150.30	748.43	158.24	772.34	333.45
	7	139.62	964.69	143.74	970.42	386.62
rat-99	2	728.71	1456.95	801.68	1603.31	442.96
	3	587.17	1751.95	636.41	1903.04	495.36
	5	469.25	2336.22	487.71	2399.87	555.85
	7	443.91	3074.30	462.59	3135.78	645.43

These results show that the proposed approached was able to converge to the near-optimal values in much less computational time, which can actually be considered running in real-time. For the MinMax costs, Table 7.19 shows the deviation errors to the optimal costs,

which presents that all errors are less than 10%. However, the more visible contribution, is that the error is actually decreasing with the increasing number of vehicles. This proves that the proposed approach is more capable of handling multiple vehicles and obtain more accurate solutions in all tested benchmarks.

Table 7.19 Deviation errors to optimal MinMax costs

Benchmark / Vehicles	**eil-51**	**berlin-52**	**eil-76**	**rat-99**
2 Vehicles	6.18%	5.00%	9.55%	10.01%
3 Vehicles	4.73%	4.41%	6.90%	8.39
5 Vehicles	4.36%	2.80%	5.29%	3.93%
7 Vehicles	4.28%	2.09%	2.95%	4.21%

Since there are several possibilities for the solutions permutations, when the MinMax solution is optimized, this does not guarantee the optimization for the total cost as well. However, the proposed approach objective function was designed to take this into consideration. Table 7.20 shows the deviation errors from the optimal total costs. It is quite obvious the same behavior of the MinMax costs, where the more vehicles are introduced to the system, the more capable the proposed approach to find near-optimal allocations. Moreover, in the case of $berlin-52$ dataset, the proposed approach obtained allocations better than the sub-optimal ones in the case of 5 and 7 vehicles.

Table 7.20 Deviation errors to optimal total costs

Benchmark / Vehicles	**eil-51**	**berlin-52**	**eil-76**	**rat-99**
2 Vehicles	6.00%	5.02%	9.31%	10.05%
3 Vehicles	4.50%	5.46%	7.56%	8.62
5 Vehicles	2.35%	**-0.95**%	3.19%	2.72%
7 Vehicles	0.18%	**-6.80**%	0.59%	2.00%

The conducted comparative study highlighted the high performance of the proposed approach in different scenarios and its scalability in handling multiple vehicles. Therefore,

the proposed approach was not only able to obtain near-optimal allocations in much less computational time, but also outperform the CPLEX approach in one of the scenarios, under the same set of constraints and conditions.

Additionally, the real-world experiment was successful in terms of allocation of the requests and the vehicles performance to pick-up and drop-off the passengers, along with the automated navigation from the starting points to the destinations. The two iCab platforms were connected via the V2V communication and once the first request was created on the webserver, the token-holder vehicle collected the requests and the available vehicles to start the proposed task allocation algorithm. The first request was allocated to iCab-1 and while the vehicle is navigating to the first destination point, the second request came in, at which the allocation algorithm output was to allocate it to iCab-1, since this optimizes the overall solution. Therefore, iCab-1 continues the navigation to the first destination point, then pick-up the second request to continue afterward to the second destination point. The last request was communicated while the vehicle is navigating and this time the allocation algorithm assigned it to iCab-2, which was free and able to pick-up the passenger. All computations are executed on-board of the vehicles computers in real-time, and since the problem was addressing 2 vehicles and 3 requests in an environment with 12 points of interests, the computational time was approx. 1860 milliseconds for each allocation solution.

Chapter 8

Conclusion and Future Work

8.1 Introduction

The efforts conducted in this thesis work are summarized in this chapter. Section 8.2 presents the conclusion as an overview about the research focus and study main points. Furthermore, Section 8.3 introduces recommendations for further research endeavors.

8.2 Conclusion

The technological advances in the Intelligent Transportation Systems (ITS) are exponentially improving over the last century. The objective is to provide intelligent and innovative services for the different modes of transportation, towards a better, safer, coordinated and smarter transport networks. The ITS focus is divided into two main categories; the first is to improve existing components of the transport networks, while the second is to develop intelligent vehicles which facilitate the transportation process.

During the last couple of decades, the majority of car manufacturers and research institutes presented different forms of vehicles with Advanced Driver Assistance Systems (ADAS) capabilities. This technological advance lead to thousands of cars today in the streets are capable of reaching the second level of automation, with both lateral and longitudinal control of the vehicle in defined use cases, taking into consideration that the driver has to monitor the system at all times and be ready to take control.

However the technological advances did not stop there, there are many vehicles on the road aiming towards the third automation level. Different research efforts have been exerted to tackle various aspects in one of the fields of the automated vehicles research. Accordingly,

this thesis addressed the problem of multiple automated vehicles cooperation and proposed an approach to solve it through various stages.

First, Unity, a 3D visualization tool with a powerful physics engine, combined with Robot Operating System (ROS) framework and Simulation of Urban Mobility (SUMO) are used to implement 3DCoAutoSim. 3DCoAutoSim is an abbreviation for "3D Simulator for Cooperative ADAS and Automated Vehicles Simulator". The study of simulation environments is essential to understand how the vehicle is going to behave before deployment in the real environment. 3DCoAutoSim was tested under different circumstances and conditions, afterward, it was validated through carrying-out several controlled experiments and compare the results against their counter reality experiments. The obtained results showed the efficiency of the simulator to handle different situations, emulating real world cars.

Second, off-road environment automated vehicles were developed as part of the iCab project. iCab is an abbreviation for "Intelligent Campus Automobile", which had two electric golf-carts that were modified mechanically, electronically and electrically towards the goal of automated driving. Each iCab was equipped with several on-board embedded computers, perception sensors and auxiliary devices, in order to execute the necessary actions for self-driving. iCab have Vehicle-to-Everything (V2X) communication schemes, three layers of control levels, cooperation architecture for multiple vehicles, several localization systems, several mapping systems, several perception systems, and finally several planning system. Hundreds of experiments were carried-out for the validation of each system in the iCab platform. Results proved the functionality of the iCab platform to self-drive from one point to another with minimal human intervention.

Furthermore, through the availability of multiple instances of the iCab platform, the problem of cooperative driving was addressed through implementing several communication schemes and studying the outcomes of platooning approach and protocols. The use of V2X communication along with a dynamic platooning approach obtained better results than vehicles that depend only on its own sensors for the environment perception.

The next objective was to design an architecture for the multiple automated vehicle coordination as Multi-robot Task Allocation (MRTA) problem, focusing on solving the shared mobility on-demand multiple transportation requests problem. The architecture was designed in a generic manner, to allow heterogeneity, scalability, adaptability, and integrability to various platforms. In order to test the proposed MRTA architecture, a hybrid optimization-based algorithm was proposed as solution method to the MRTA problem. The novel approach utilize both Simulated Annealing (SA) and Genetic Algorithm (GA) approaches, in order to allocate the vehicles in a near-optimal solution, optimizing the distance covered, execution time, waiting time and vehicle energy. The proposed algorithm was tested against several

benchmarks for validation of obtaining near-optimal solutions. The results showed high performance of the proposed algorithm in tackling several form of the MRTA problem and its scalability in handling multiple vehicles, outperforming all other approaches.

Finally, the work in this thesis derived 34 peer-reviewed publications in books, journals and conferences, including 2 best paper awards, which are listed and detailed in Appendix A.

8.3 Future Work

While the proposed work obtained successful results, there were some aspects of the problem that were assumed for simplification. These assumptions could be addressed in the future work, to increase the reliability of the proposed work and to incorporate the proposed algorithms in a different types of vehicles, aiming towards a complete self-driving platform. The future recommendations are, but are not limited to:

- Extending the proposed approaches of the iCab platforms to different type of automated vehicles. Accordingly, all the proposed systems will be adapted to the new automated car and later to be tested in real urban and high-way environments.
- Extending the proposed V2X communication schemes to adapt to the environment networks availability and utilize 5G networks. Therefore, the communication speed is increased among all road entities, and provide the possibility of near real-time information sharing and more efficient communication system.
- Extending the networking feature of the proposed 3DCoAutoSim driving simulator to allow multiple user controlled vehicles in the environment along with automated vehicles. Furthermore, carry out experiments for the intelligent vehicles approaches with guaranteed ground-truth.
- The vehicle platooning approach assumed that all vehicles have to follow the same trajectory during the whole trip. However, for a more generic approach, protocols of joining and leaving the platoon are to be investigated and with multiple vehicles in a distributed manner, thus each vehicle has a local leader to follow.
- Extending the proposed MRTA cooperation architecture heterogeneity, scalability, and adaptability features to carry-out more experiments that include transportation requests for both passengers and packages, using iCab ground platforms and SkyOnyx aerial platforms respectively. Furthermore, the experiments should be carried-out in different off-road unknown environments.

- In this thesis, the complex tasks decomposition was executed analytically based on the number of passengers for each request and vehicle capacity. However, for the aim of complete autonomy of the system, further research to propose an algorithm that is responsible for the decomposition of the complex tasks based on the objective function is required. This ensure the adaptability of the proposed architecture for heterogeneous vehicles and/or tasks.

Appendices

Appendix A

Publications List

Book Chapters

1. **Ahmed Hussein**, Fernando Garcia, and Cristina Olaverri-Monreal (2018). "Hands-on Tutorial: How to use Unity as Simulator for ROS-based Applications". Chapter in Robot Operating System (ROS), Vol. 4, pp. 1-22. Springer International Publishing. *Under-Review*.

2. Abdulla Al-Kaff, Francisco Miguel Moreno and **Ahmed Hussein** (2018). "ROS-based Approach for Unmanned Vehicles in Civil Applications". Chapter in Robot Operating System (ROS), Vol. 3, pp. 1-30. Springer International Publishing.

3. David Martin, Pablo Marin-Plaza, **Ahmed Hussein**, Arturo de la Escalera and Jose Maria Armingol (2016). "ROS-based Architecture for Autonomous Intelligent Campus Automobile (iCab)". Chapter in UNED Plasencia Revista de Investigacion Universitaria, Vol. 12, pp. 257-272. Agbatanero.

4. Alaa Khamis, **Ahmed Hussein** and Ahmed Elmogy (2015). "Multi-robot Task Allocation: A Review of the State-of-the-art". Chapter in Cooperative Robots and Sensor Networks, Vol. 604, pp. 31-51. Springer International Publishing.

Journal Papers

1. **Ahmed Hussein**, Francisco Miguel Moreno, Fernando Garcia, and Jose Maria Armingol (2018). "Multiple Vehicle Cooperation Generic ROS-based Architecture for Transportation Requests". In Integrated Computer-Aided Engineering, pp. 1-16. IOS Press. *Under-Review*.

2. Pablo Marin-Plaza, **Ahmed Hussein**, David Martin, and Arturo de la Escalera (2018). "iCab Use Case for ROS-based Architecture". In Iberian Robotics (ROBOT2018), pp. 1-31. Springer. *Under-Review*.

3. Aso Validi, Thomas Ludwig, **Ahmed Hussein** and Cristina Olaverri-Monreal (2018). "Impact of different penetration rates of Vehicle-to-Vehicle Communication and Adaptive Cruise Control on road safety by means of simulated environments". In Intelligent Transportation Systems Magazine, pp. 1-9. IEEE. *Under-Review*.

4. **Ahmed Hussein**, Pablo Marin-Plaza, Fernando Garcia, and Jose Maria Armingol (2018). "Hybrid optimization-based approach for multiple intelligent vehicles requests allocation". In Journal of Advanced Transportation, pp. 1-12. Hindawi Publishing Corp.

5. Pablo Marin-Plaza, **Ahmed Hussein**, David Martin, and Arturo de la Escalera (2018). "Global and Local Path Planning Study in a ROS-based Research Platform for Autonomous Vehicles". In Journal of Advanced Transportation, pp. 1-11. Hindawi Publishing Corp.

6. Pablo Marin-Plaza, **Ahmed Hussein**, David Martin, and Arturo de la Escalera (2017). "Complete ROS-based Architecture for Intelligent Vehicles". In Iberian Robotics (ROBOT2017), pp. 499-510. Springer International Publishing.

7. **Ahmed Hussein**, Mohamed Adel, Mohamed Bakr, Omar M. Shehata, and Alaa Khamis (2014). "Multi-Robot Task Allocation for Search and Rescue Missions". In Journal of Physics: Conference Series, Vol. 570, No. 5, pp. 1-10. IOP Publishing.

8. Mohamed Badreldin, **Ahmed Hussein**, and Alaa Khamis (2013). "A comparative study between optimization and market-based approaches to multi-robot task allocation". In Journal of Advances in Artificial Intelligence, pp. 1-12. Hindawi Publishing Corp.

Conference Papers

1. **Ahmed Hussein**, Alberto Diaz-Alvarez, Jose Maria Armingol, and Cristina Olaverri-Monreal (2018). "3DCoAutoSim: Simulator for Cooperative ADAS and Automated Vehicles". In proceedings of International Conference on Intelligent Transportation Systems (ITSC2018), pp. 1-6. IEEE. *Under-Review*.

2. Abdulla Al-Kaff, Ricardo Alonso, Mostafa Osman, and **Ahmed Hussein** (2018). "SkyOnyx: Autonomous UAV Research Platform for Air Transportation System (ATSys)". In proceedings of International Conference on Intelligent Transportation Systems (ITSC2018), pp. 1-6. IEEE. *Under-Review*.

3. Mostafa Osman, Ricardo Alonso, Francisco Miguel Moreno, **Ahmed Hussein**, and Abdulla Al-Kaff (2018). "Extended H-Infinity Filter for Multi-Sensor Fusion Localization". In proceedings of International Conference on Intelligent Transportation Systems (ITSC2018), pp. 1-6. IEEE. *Under-Review*.

4. Ayman H. Abdelhamid, Miguel Angel de Miguel, Pablo Marin Plaza, **Ahmed Hussein**, and Fernando Garcia (2018). "Model Predictive Linear Quadratic Regulator Based Path Tracking for Automated Ground Vehicles". In proceedings of International Conference on Intelligent Transportation Systems (ITSC2018), pp. 1-6. IEEE. *Under-Review*.

5. Ahmed H. Salem, Catherine M. Elias, Omar M. Shehata, Elsayed I. Morgan, and **Ahmed Hussein** (2018). "Two-Stage Voronoi-Based Deployment Algorithm with Obstacle Avoidance for 2D CEAC Control Problem". In proceedings of International Conference on Intelligent Transportation Systems (ITSC2018), pp. 1-6. IEEE. *Under-Review*.

6. Dalia M. Ibrahim, Catherine M. Elias, Omar M. Shehata, Elsayed I. Morgan, and **Ahmed Hussein** (2018). "On-line Collision-Free Path Planning Algorithm using Various Artificial Potential Field Approaches". In proceedings of International Conference on Intelligent Transportation Systems (ITSC2018), pp. 1-6. IEEE. *Under-Review*.

7. Raul Sosa San Frutos, **Ahmed Hussein**, Abdulla Al Kaff and Arturo de La Escalera (2018). "ROS-based Architecture for Multiple Robots Formation". In proceedings of International Conference on Intelligent Robots and Systems (IROS2018), pp. 1-6. IEEE. *Under-Review*.

8. Mostafa Osman, **Ahmed Hussein**, Abdulla Al-Kaff, Fernando Garcia and Jose Maria Armingol (2018). "Online Adaptive Covariance Estimation Approach for Multiple

Odometry Sensors Fusion". In proceedings of International Vehicles Symposium (IV2018), pp. 1-6. IEEE.

9. Omar M. Shehata, **Ahmed Hussein**, Mohamed Abdelaziz and Farid Tolbah (2018). "ATOM: Autonomous Transportation Operating Modules - A Framework for Autonomous Vehicles Development". In proceedings of International Vehicles Symposium (IV2018), pp. 1-6. IEEE.

10. **Ahmed Hussein**, Pablo Marin-Plaza, Fernando Garcia and Jose Maria Armingol (2017). "Autonomous Cooperative Driving Using V2X Communications in Off-Road Environment". In proceedings of International Conference on Intelligent Transportation Systems (ITSC2017), pp. 1-6. IEEE.

11. Noelia Hernandez, **Ahmed Hussein**, Daniel Cruzado, Ignacio Parra, and Jose Maria Armingol (2017). "Applying Low Cost WiFi-based Localization to In-Campus Autonomous Vehicles". In proceedings of International Conference on Intelligent Transportation Systems (ITSC2017), pp. 1-6. IEEE.

12. Armando Astudillo, Francisco Miguel Moreno, **Ahmed Hussein**, and Fernando Garcia (2017). "Cost-Efficient Brainwave Controller for Automated Vehicles Route Decisions". In proceedings of International Conference on Intelligent Transportation Systems (ITSC2017), pp. 51-56. IEEE.

13. Andras Kokuti, **Ahmed Hussein**, Arturo de la Escalera, and Fernando Garcia (2017). "Market-based Approach for Cooperation and Coordination Among Multiple Autonomous Vehicles". In proceedings of International Conference on Intelligent Transportation Systems (ITSC2017), pp. 1-6. IEEE.

14. Andras Kokuti, **Ahmed Hussein**, Pablo Marin-Plaza, Arturo de la Escalera, and Fernando Garcia (2017). "V2X Communications Architecture for Off-Road Autonomous Vehicles". In proceedings of International Conference on Vehicular Electronics and Safety (ICVES2017), pp. 69-74. IEEE.

15. Arman Allamehzadeh, Jesus Urdiales de la Parra, **Ahmed Hussein**, Fernando Garcia, and Cristina Olaverri-Monreal (2017). "Cost-efficient driver state and road conditions monitoring system for conditional automation". In proceedings of Intelligent Vehicles Symposium (IV2017), pp. 1497-1502. IEEE.

16. Pablo Marin-Plaza, **Ahmed Hussein**, David Martin, Fernando Garcia, Arturo de la Escalera and Jose Maria Armingol (2016). "ROS-based Architecture for Autonomous

Vehicles". In proceedings of Symposium SEGVAUTO-TRIES-CM: Technologies for a Safe, Accessible and Sustainable Mobility, pp. 43-46. UPM.

17. **Ahmed Hussein**, Fernando Garcia, Jose Maria Armingol and Cristina Olaverri-Monreal (2016). "P2V and V2P Communication for Pedestrian Warning on the basis of Autonomous Vehicles". In proceedings of International Conference on Intelligent Transportation Systems (ITSC2016), pp. 2034-2039. IEEE.

18. Pablo Marin-Plaza, **Ahmed Hussein**, Carlos Guindel, Fernando Garcia, David Martin and Arturo de la Escalera (2016). "Arquitectura basada en ROS para el vehiculo iCab (Intelligent Campus Automobile)". In proceedings of XXXVII Jornadas de Automatica, pp. 531-536. UNED.

19. **Ahmed Hussein**, Pablo Marin-Plaza, David Martin, Arturo de la Escalera and Jose Maria Armingol (2016). "Autonomous Off-Road Navigation using Stereo-Vision and Laser-Rangefinder Fusion for Outdoor Obstacles Detection". In proceedings of Intelligent Vehicles Symposium (IV2016), pp. 104-109. IEEE.

20. Pablo Marin-Plaza, Jorge Beltran, **Ahmed Hussein**, David Martin, Arturo de la Escalera and Jose Maria Armingol (2016). "Stereo Vision-based Local Occupancy Grid Map for Autonomous Navigation in ROS". In proceedings of International Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP2016), Vol. 3, pp. 701-706. SCITEPRESS.

21. **Ahmed Hussein**, Abdulla Al-Kaff, Arturo de la Escalera and Jose Maria Armingol (2015). "Autonomous Indoor Navigation of Low-Cost Quadcopters". In proceedings of International Conference on Service Operations and Logistics, and Informatics (SOLI2015), pp. 133-138. IEEE.

Abstract Papers

1. **Ahmed Hussein**, Pablo Marin-Plaza, Fernando Garcia, and Jose Maria Armingol (2017). "Optimization-based Approach for Cooperation and Coordination of Multi-Autonomous Vehicles". In proceedings of Sixteenth International Conference on Computer Aided Systems Theory (EUROCAST2017), pp. 1-2. Springer.

Awards

1. **Best Computer Vision Award** (Co-author) in XXXVII Jornadas de Automatica 2016.
2. **Best Student Paper Award** in IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI2015).

Appendix B

Theses Supervision

2018

- Miguel Descalzo (2018). *Pedestrian Cross Detection for Autonomous Vehicles*. Universidad Carlos III de Madrid, Bachelor Thesis.
- Ayman Abdelhamid (2018). *Automatic Navigation Control for Autonomous Unmanned Ground Vehicles*. Universidad Carlos III de Madrid, Ain Shams University Bachelor Thesis Abroad.
- Ahmed Nasr (2018). *Weight Adaptive Braking Control for Autonomous Ground Vehicles*. Universidad Carlos III de Madrid, Ain Shams University Bachelor Thesis Abroad.
- Elnagdy Hisham (2018). *Modeling and Control of Steer by Wire System*. Universidad Carlos III de Madrid, Ain Shams University Bachelor Thesis Abroad.
- Amr Ramadan (2018). *ROS-based Advanced Platooning Approach for Automated Vehicles*. Universidad Carlos III de Madrid, German University in Cairo Bachelor Thesis Abroad.
- Osama Ahmed (2018). *Generating and Analyzing Occupancy Grid Maps for Intelligent Vehicles*. Universidad Carlos III de Madrid, German University in Cairo Bachelor Thesis Abroad.
- Yahia (2018). *Accelerated Detection and Localization for Unmanned Aerial Vehicles*. Universidad Carlos III de Madrid, German University in Cairo Bachelor Thesis Abroad.

- Mohammad Abdeen (2018). *Trajectory Velocity Planning and Control for Automated Vehicles*. Universidad Carlos III de Madrid, German University in Cairo Bachelor Thesis Abroad.
- Mohamed El Tabei (2018). *Hexarotor Drone Stability Control Using Flaps*. Universidad Carlos III de Madrid, German University in Cairo Bachelor Thesis Abroad.
- Ahmed Elghobashy (2018). *Scene Understanding Visualization for Automated Vehicles using Unity3D*. Universidad Carlos III de Madrid, German University in Cairo Bachelor Thesis Abroad.

2017

- Armando Astudillo (2017). *Brainwave Controller Development For Intelligent Vehicle*. Universidad Carlos III de Madrid, Bachelor Thesis.
- Alvaro Huete (2017). *Design and implementation of a brake controller for an intelligent vehicle (iCab)*. Universidad Carlos III de Madrid, Bachelor Thesis.
- Irene Salazar (2017). *Mobile Warning Application*. Universidad Carlos III de Madrid, Bachelor Thesis.
- Yekaterina Lertxundi (2017). *Steering Controller for Intelligent Vehicle*. Universidad Carlos III de Madrid, Bachelor Thesis.
- Ahmed Fahmy (2017). *EKF-based Multi-sensor Fusion Localization for Unmanned Ground Vehicles*. Universidad Carlos III de Madrid, German University in Cairo Bachelor Thesis Abroad.
- Abdallah Abdelwahed (2017). *UKF-based Multi-sensor Fusion Localization for Unmanned Aerial Vehicles*. Universidad Carlos III de Madrid, German University in Cairo Bachelor Thesis Abroad.

2016

- Omar Fathi (2016). *Design and Test Wireless Communication Protocol for Autonomous Vehicles*. Universidad Carlos III de Madrid, German University in Cairo Bachelor Thesis Abroad.

- Ahmed Radwan (2016). *Design and Test a Control System for a Quadcopter*. Universidad Carlos III de Madrid, German University in Cairo Bachelor Thesis Abroad.

2015

- Ossama El Hammady (2015). *Outdoors Odometry and Mapping for Autonomous Golf Carts*. Universidad Carlos III de Madrid, German University in Cairo Bachelor Thesis Abroad.
- Hesham Nounou (2015). *Vision-based Surveillance System for Counting People Using Small Quadcopter*. Universidad Carlos III de Madrid, German University in Cairo Bachelor Thesis Abroad.

Appendix C

3DCoAutoSim Read-Me

Brief Description

This is the Read-Me manual for the 3D-CoAutoSim, which is an abbreviation for "3D Cooperative Automated Vehicles Simulator", which has VANET capabilities and connected to SUMO and ROS. The manual details the simulator functionalities, capabilities description, in addition to how-to guides for future modifications and improvements.

This is for **3D-CoAutoSim v4.0**, which was implemented using:

- Unity 2018.1.0b12
- SUMO 0.32.0
- ROS Kinetic Kame

In order to run the simulator, your system must be:

- OS: Windows 7 or higher
- CPU: Minimum quad-core with 2.6GHz
- GPU: Minimum Nvidia GTX 940
- RAM: Minimum 8GB

How to run?

The simulator runs by double clicking on the built executable file, and the configuration window appears, screenshot is shown in Figure C.1. Through this window you can adjust

the graphics settings from the **Graphics** tab, by selecting screen resolution (default is 1920x1080), graphics quality (default is ultra), display monitor, and whether you want the application to be Windows or Full-Screen (default). Moreover, you can go to the **Input** tab and remap the input buttons to your controller. Upon finalizing all settings, click **Play** button and you will be redirected to the simulator configuration menu.

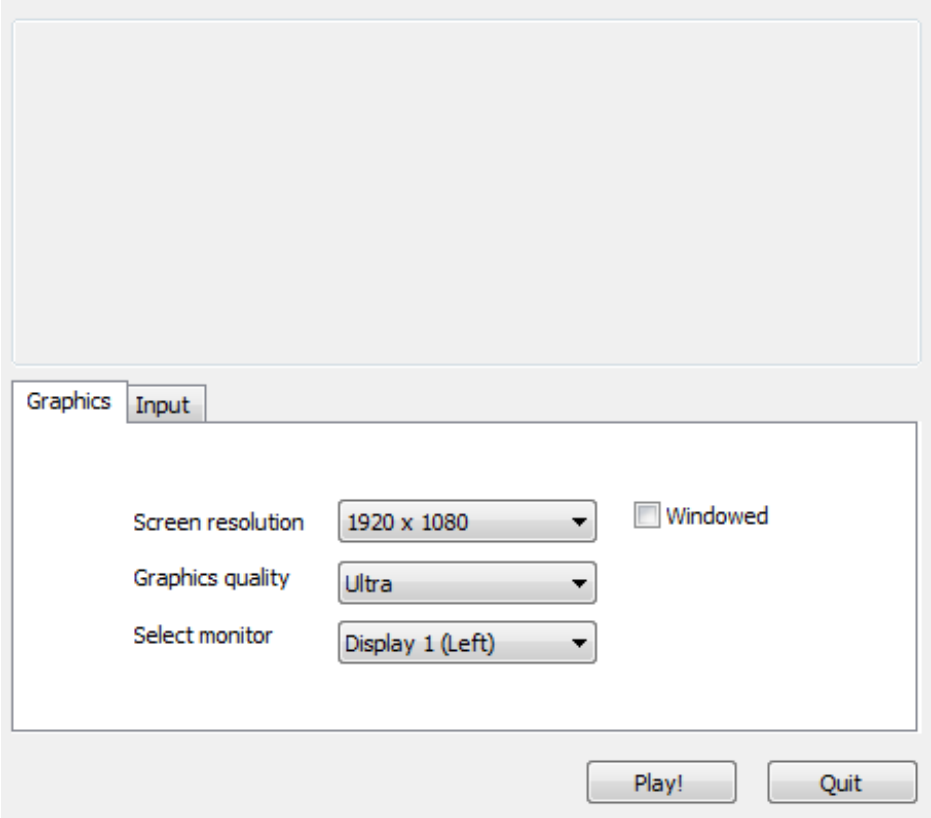


Figure C.1 Configuration window

A screenshot of the simulator configuration menu is shown in Figure C.2, through which you can select the desired options from the available five categories, in the following order. First category is the environments, where you must select one, and only one, environment for your simulator to build and add the vehicle to it. Second category is the scenarios, where in case a specific scenario is defined for experiment, select it instead of the environment. Third category is the features, where you select as many features as needed to be incorporated to the vehicle. Fourth category is the devices, each selected device will be mounted on the vehicle and enabled for publishing data. Automatically, last category is the outputs, where they are selected according to which device is mounted or vice versa, to record the data in the **Output** folder.

The output files format is adjusted based on the buttons on the right-side, currently only **CSV** format is available. At the top-right, the simulator mode can be adjusted to work either in Single or Multiple simulator modes. Moreover, for the ROS connectivity, click the **ROS** button on the right side to enable/disable it.

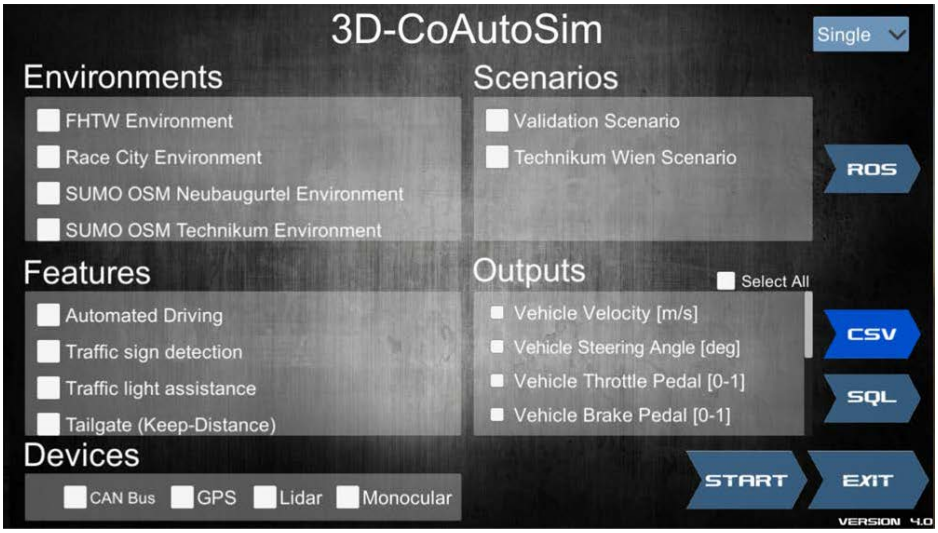


Figure C.2 Simulator configuration menu

Upon finalizing all configurations, you click on **Start** button, which will load the simulation environment with your configuration, ready to drive with the controller. Once the driving experiment ends, press **ESC** button to exit the application or return to the configuration menu. A screenshot example of the simulation environment is show in Figure C.3.

Controller Description

The vehicle can be controlled with both Keyboard and Joystick, the buttons mapping is listed below, which can be configured from the "**RCC_CarControllerV3**" script, located in "**/Assets/ExternalAssets/RealisticCarControllerV3/Scripts/**" folder.

Keyboard

The following buttons can be found on a standard 104-key computer keyboard, the letter buttons don't have to be in capital mode.

- **Up-arrow** or **W**: increases the speed of the vehicle, acts like a throttle

Figure C.3 Simulator simulation environment

- **Down-arrow** or **S**: decreases the speed of the vehicle, acts like a brake
- **Right-arrow** or **D**: rotates the vehicle steering to the right
- **Left-arrow** or **A**: rotates the vehicle steering to the left
- **1,2,3,4,5,6,7**: shifts vehicle gear to the pressed number, acts like shifter, active only in manual driving mode
- **C**: cycles camera views as per the allocated cameras in the vehicle (first-person camera, third-person camera, orbit camera, top-down camera, & wheel camera)
- **I**: activates/deactivates the vehicle ignition engine
- **M**: cycles the driving modes as per the allocated modes in the vehicle (automatic, semi-automatic, & manual)
- **R**: shifts vehicle gear to rear, thus throttle speed drives backward, active in manual or semi-automatic driving mode
- **Left-Shift**: shifts vehicle one gear up, active only in semi-automatic mode
- **Left-Control**: shifts vehicle one gear down, active only in semi-automatic mode
- **E**: activates/deactivates the vehicle right-side blinker
- **Q**: activates/deactivates the vehicle left-side blinker

- **L**: activates/deactivates the vehicle driving front-light beam
- **K**: activates the vehicle high front-light beam, acts like light blinker

Joystick

The following buttons can be found on the Thrustmaster TH8A pedals and shifter, and the Thrustmaster T500 RS steering wheel, as shown in Figure C.4.

Figure C.4 Thrustmaster TH8A Pedals & Shifter + T500 RS Steering Wheel

Please refer to Figure C.5 for the buttons mapping.

- **Throttle-Pedal**: increases the speed of the vehicle
- **Brake-Pedal**: decreases the speed of the vehicle
- **Steering-Wheel**: rotates the vehicle steering to the left or right
- **Gear-Shifter**: shifts vehicle gear to the selected number, active only in manual driving mode, and gear number 8 represents rear gear
- **PS**: cycles camera views as per the allocated cameras in the vehicle (first-person camera, third-person camera, orbit camera, top-down camera, & wheel camera)
- **Start**: activates/deactivates the vehicle ignition engine

- **Select**: cycles the driving modes as per the allocated modes in the vehicle (automatic, semi-automatic, & manual)
- **R1**: shifts vehicle one gear up, active only in semi-automatic mode
- **L1**: shifts vehicle one gear down, active only in semi-automatic mode
- **R2**: activates/deactivates the vehicle right-side blinker
- **L2**: activates/deactivates the vehicle left-side blinker
- Δ: activates/deactivates the vehicle driving front-light beam
- *X*: activates the vehicle high front-light beam, acts like light blinker

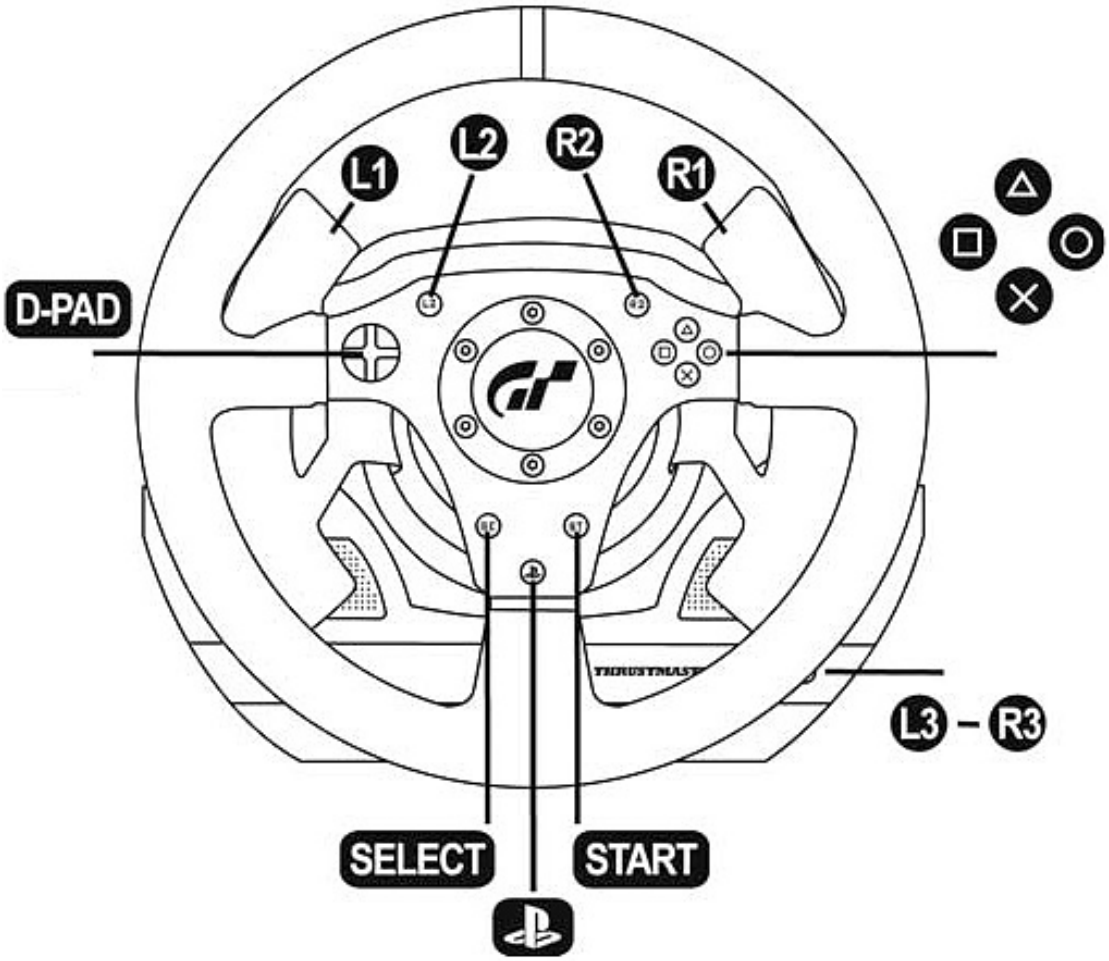


Figure C.5 Thrustmaster T500 RS Steering Wheel Buttons Mapping

Bibliography

[1] K. Benz, "Vehicle with gas engine operation," *German Patent Number DRP-37435*, 1886.

[2] A. Jambor and M. Beyer, "New cars - new materials," *Materials & design*, vol. 18, no. 1-6, pp. 203–209, 1997.

[3] K. Yamane and S. Furuhama, "A study on the effect of the total weight of fuel and fuel tank on the driving performances of cars," *International journal of hydrogen energy*, vol. 23, no. 9, pp. 825–831, 1998.

[4] Y. Li, Z. Lin, A. Jiang, and G. Chen, "Use of high strength steel sheet for lightweight and crashworthy car body," *Materials & design*, vol. 24, no. 3, pp. 177–182, 2003.

[5] W. J. Joost, "Reducing vehicle weight and improving us energy efficiency using integrated computational materials engineering," *Jom*, vol. 64, no. 9, pp. 1032–1038, 2012.

[6] F. R. Field III, T. J. Wallington, M. Everson, and R. E. Kirchain, "Strategic materials in the automobile: a comprehensive assessment of strategic and minor metals use in passenger cars and light trucks," *Environmental science & technology*, vol. 51, no. 24, pp. 14 436–14 444, 2017.

[7] M.-P. Todor, C. Bulei, and I. Kiss, "An overview on fiber-reinforced composites used in the automotive industry," *Annals of the Faculty of Engineering Hunedoara*, vol. 15, no. 2, pp. 1–181, 2017.

[8] R. Cuerden, L. Lloyd, C. Wallbank, and M. Seidl, "The potential for vehicle safety standards to prevent road deaths and injuries," *Global NCAP*, pp. 1–63, 2015.

[9] A. Paul, R. Chauhan, R. Srivastava, and M. Baruah, "Advanced driver assistance systems," SAE Technical Paper, Tech. Rep., 2016.

[10] W. D. Jones, "Building safer cars," *IEEE Spectrum*, vol. 39, no. 1, pp. 82–85, 2002.

[11] C. Castro and T. Horberry, *The human factors of transport signs*. CRC press, 2004.

[12] C. Ho and C. Spence, "Human factors in road and rail transport," *Global NCAP*, pp. 1–48, 2008.

[13] S. G. Klauer, F. Guo, B. G. Simons-Morton, M. C. Ouimet, S. E. Lee, and T. A. Dingus, "Distracted driving and risk of road crashes among novice and experienced drivers," *New England journal of medicine*, vol. 370, no. 1, pp. 5–59, 2014.

[14] A. Ghosh, T. Chatterjee, S. Samanta, J. Aich, and S. Roy, "Distracted driving: A novel approach towards accident prevention," *Advances in Computational Sciences and Technology*, vol. 10, no. 8, pp. 269–2705, 2017.

[15] WHO, *Global status report on road safety*. World Health Organization, 2015.

[16] S. Barbieri, G. Vettore, V. Pietrantonio, R. Snenghi, A. Tredese, M. Bergamini, S. Previato, A. Stefanati, R. M. Gaudio, and P. Feltracco, "Pedestrian inattention blindness while playing pokemon go as an emerging health-risk behavior: a case report," *Journal of medical internet research*, vol. 19, no. 4, 2017.

[17] B. W. Smith, "Sae levels of driving automation," *Center for Internet and Society, Stanford Law School*, 2013.

[18] R. R. Teetor, "Speed control device for resisting operation of the accelerator," 1950, uS Patent 2,519,859.

[19] P. I. Labuhn and W. J. Chundrlik Jr, "Adaptive cruise control," 1995, uS Patent 5,454,442.

[20] H. W. Kim, "Lane keeping assist system," 2012, uS Patent 8,095,266.

[21] R. Bradley, "Tesla autopilot, the electric-vehicle maker sent its cars a software update that suddenly made autonomous driving a reality," 2016.

[22] I. Schadler, "Eco-mobility 2025 plus - roadmap," *A3PS*, pp. 1–72, 2015.

[23] C. Benevolo, R. P. Dameri, and B. D'Auria, "Smart mobility in smart city," *Empowering Organizations*, pp. 13–28, 2016.

[24] S. Shaheen, A. Cohen, and I. Zohdy, "Shared mobility: current practices and guiding principles," *U.S. Department of Transportation*, pp. 1–120, 2016.

[25] V. Dolk, J. den Ouden, S. Steeghs, J. G. Devanesan, I. Badshah, A. Sudhakaran, K. Elferink, and D. Chakraborty, "Cooperative automated driving for various traffic scenarios: Experimental validation in the gcdc 2016," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1308–1321, 2018.

[26] P. R. Alves, J. Goncalves, R. J. Rossetti, E. C. Oliveira, and C. Olaverri-Monreal, "Forward collision warning systems using heads-up displays: Testing usability of two new metaphors," in *Intelligent Vehicles Symposium (IV) Workshop*, 2013, pp. 1–6.

[27] C. Olaverri-Monreal, P. Gomes, M. K. Silveria, and M. Ferreira, "In-vehicle virtual traffic lights: a graphical user interface," in *Information Systems and Technologies (CISTI), 2012 7th Iberian Conference on*. IEEE, 2012, pp. 1–6.

[28] R. Maia, M. Silva, R. Araujo, and U. Nunes, "Electric vehicle simulator for energy consumption studies in electric mobility systems," in *Forum on Integrated and Sustainable Transportation System (FISTS)*. IEEE, 2011, pp. 227–232.

[29] A. Wegener, M. Piorkowski, M. Raya, H. Hellbruck, S. Fischer, and J.-P. Hubaux, "Traci: an interface for coupling road traffic and network simulators," in *Proceedings of the 11th communications and networking simulation symposium*. ACM, 2008, pp. 155–163.

[30] NSNAM, "ns-2: The network simulator," [Last Accessed: 2018-04-30]. [Online]. Available: http://nsnam.sourceforge.net/wiki/index.php/Main_Page

[31] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo (simulation of urban mobility): an overview," *The Third International Conference on Advances in System Simulation (SIMUL)*, 2011.

[32] C. Sommer, R. German, and F. Dressler, "Bidirectionally coupled network and road traffic simulation for improved ivc analysis," *IEEE Transactions on Mobile Computing*, vol. 10, no. 1, pp. 3–15, 2011.

[33] P. Bhavsar, M. Chowdhury, Y. He, and M. Rahman, "A network wide simulation strategy of alternative fuel vehicles," *Transportation Research Part C: Emerging Technologies*, vol. 40, pp. 201–214, 2014.

[34] A. Brown *et al.*, "Udacity self-driving car simulator," [Last Accessed: 2018-04-30]. [Online]. Available: https://github.com/udacity/self-driving-car-sim

[35] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[36] R. Math, A. Mahr, M. M. Moniri, and C. Muller, "Opends: A new open-source driving simulator for research," in *Adjunct Proceedings of the 4th International Conference on Automotive User Interfaces and Interactive Vehicular Appilcations, Portsmouth, NH, USA*, 2012, pp. 7–8.

[37] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," *ICRA workshop on open source software*, pp. 1–5, 2009.

[38] D. Helgason, "Unity 1.0 is shipping," [Last Accessed: 2018-04-30]. [Online]. Available: https://forum.unity.com/threads/unity-1-0-is-shipping.56/

[39] A. B. Lange, U. P. Schultz, and A. S. Soerensen, "Unity-link: A software-gateware interface for rapid prototyping of experimental robot controllers on fpgas," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2013, pp. 3899–3906.

[40] R. Codd-Downey, P. M. Forooshani, A. Speers, H. Wang, and M. Jenkin, "From ros to unity: Leveraging robot and virtual environment middleware for immersive teleoperation," in *IEEE International Conference on Information and Automation (ICIA)*. IEEE, 2014, pp. 932–936.

[41] R. T. Jonathan Mace and J. Lee, "Ros bridge suite," [Last Accessed: 2018-04-30]. [Online]. Available: http://wiki.ros.org/rosbridge_suite

[42] D. Crockford, "The application/json media type for javascript object notation (json)," *IETF*, pp. 1–10, 2006.

[43] W. Meng, Y. Hu, J. Lin, F. Lin, and R. Teo, "Ros + unity: An efficient high-fidelity 3d multi-uav navigation and control simulator in gps-denied environments," in *41st Annual Conference of the IEEE Industrial Electronics Society (IECON)*. IEEE, 2015, pp. 2562–2567.

[44] Y. Hu and W. Meng, "Rosunitysim: Development and experimentation of a real-time simulator for multi-unmanned aerial vehicle local planning," *Simulation*, vol. 92, no. 10, pp. 931–944, 2016.

[45] E. Sita, C. M. Horvath, T. Thomessen, P. Korondi, and A. G. Pipe, "Ros-unity3d based system for monitoring of an industrial robotic process," in *IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2017, pp. 1047–1052.

[46] Y. Mizuchi and T. Inamura, "Cloud-based multimodal human-robot interaction simulator utilizing ros and unity frameworks," in *IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2017, pp. 948–955.

[47] M. Jenkin, "Unity ros," [Last Accessed: 2018-04-30]. [Online]. Available: https://github.com/michaeljenkin/unityros

[48] E. Ackerman, "Turtlebot inventors tell us everything about the robot," *IEEE Spectrum*, pp. 1–10, 2013.

[49] M. C. Thorstensen, "Visualization of robotic sensor data with augmented reality," Master's thesis, Universitetet i Oslo, 2017.

[50] M. C. Thorstensen, "Ros bridge lib," [Last Accessed: 2018-04-30]. [Online]. Available: https://github.com/MathiasCiarlo/ROSBridgeLib

[51] S. Kato, S. Tsugawa, K. Tokuda, T. Matsui, and H. Fujii, "Vehicle control algorithms for cooperative driving with automated vehicles and intervehicle communications," *IEEE Transaction on Intelligent Transportation Systems*, vol. 3, pp. 155–161, 2002.

[52] J. Jiong, J. Gubbi, S. Marusic, and M. Palaniswami, "An information framework for creating a smart city through internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 2, pp. 112–121, 2014.

[53] D. L. Bock, D. Kettles, and J. Harrison, "Automated, autonomous and connected vehicle technology assessment," *Electric Vehicle Transportation Center (EVTC)*, 2016.

[54] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[55] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[56] S. L. Poczter and L. M. Jankovic, "The google car: driving toward a better future," *Journal of Business Case Studies*, vol. 10, no. 1, pp. 1–7, 2014.

[57] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller *et al.*, "Making bertha drive—an autonomous journey on a historic route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, 2014.

[58] P. Furgale, U. Schwesinger, M. Rufli, W. Derendarz, H. Grimmett, P. Mühlfellner, S. Wonneberger, J. Timpner, S. Rottmann, B. Li, B. Schmidt, T. N. Nguyen, E. Cardarelli, S. Cattani, S. Brüning, S. Horstmann, M. Stellmacher, H. Mielenz, K. Köser, M. Beermann, C. Häne, L. Heng, G. H. Lee, F. Fraundorfer, R. Iser, R. Triebel, I. Posner, P. Newman, L. Wolf, M. Pollefeys, S. Brosig, J. Effertz, C. Pradalier, and R. Siegwart, "Toward automated driving in cities using close-to-market sensors, an overview of the v-charge project," *IEEE Intelligent Vehicles Symposium (IV)*, pp. 809–816, 2013.

[59] DailyNews, "Bmw, audi push self-driving cars closer to reality," *Daily News - Autos*, 2013.

[60] A. Bragman, "Mercedes-benz tech brings cars closer to self driving," *USA Today*, 2016.

[61] L. Laursen, "Volvo to test self-driving cars in traffic," *IEEE Spectrum - Tech Talk*, 2013.

[62] A. Y. S. Lam, Y.-W. Leung, and X. Chu, "Autonomous vehicle public transportation system," *IEEE International Conference Connected Vehicles and Expo (ICCVE)*, p. 571–576, 2014.

[63] P. Khaligh and U. Weidmann, "A conceptual framework for the interactions of autonomous public transport systems and urban planning guideline," *Swiss Transport Research Conference*, pp. 1–18, 2016.

[64] M. M. Waldrop *et al.*, "No drivers required," *Nature*, vol. 518, no. 7537, p. 20, 2015.

[65] P. Parsania and K. Saradava, "Drive-by-wire systems in automobiles," *Journal of Systematic Computing*, vol. 6, pp. 1–5, 2012.

[66] A. Kader, "Steer-by-wire control system," Ph.D. dissertation, Swarthmore College Department of Engineering, 2006.

[67] R. Isermann, R. Schwarz, and S. Stolzl, "Fault-tolerant drive-by-wire systems," *IEEE Control Systems*, vol. 22, no. 5, pp. 64–81, 2002.

[68] W. Xiang, P. C. Richardson, C. Zhao, and S. Mohammad, "Automobile brake-by-wire control system design and analysis," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 1, pp. 138–145, 2008.

[69] P. Papadimitratos, A. D. L. Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza, "Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation," *IEEE Communications Magazine*, vol. 47, no. 11, pp. 84–95, 2009.

[70] P. Wang, B. Di, H. Zhang, K. Bian, and L. Song, "Cellular v2x in unlicensed spectrum: Harmonious coexistence with vanet in 5g systems," *arXiv preprint arXiv:1712.04639*, pp. 1–31, 2017.

[71] H. Hartenstein and K. Laberteaux, *VANET: vehicular applications and inter-networking technologies*. John Wiley & Sons, 2009, vol. 1.

[72] H. M. Fahmy, G. Baumann, M. A. A. El Ghany, and H. Mostafa, "V2v-based vehicle risk assessment and control for lane-keeping and collision avoidance," in *29th International Conference on Microelectronics (ICM2017)*. IEEE, 2017, pp. 1–5.

[73] N. Liu, M. Liu, J. Cao, G. Chen, and W. Lou, "When transportation meets communication: V2p over vanets," in *IEEE International Conference on Distributed Computing Systems (ICDCS2010)*. IEEE, 2010, pp. 567–576.

[74] W. Cho, S. I. Kim, H. kyun Choi, H. S. Oh, and D. Y. Kwak, "Performance evaluation of v2v/v2i communications: The effect of midamble insertion," in *International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace and Electronic Systems Technology*. IEEE, 2009, pp. 793–797.

[75] I. C. Msadaa, P. Cataldi, and F. Filali, "A comparative study between 802.11 p and mobile wimax-based v2i communication networks," in *Fourth International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST2010)*. IEEE, 2010, pp. 186–191.

[76] G. Cotugno, L. D'Alfonso, W. Lucia, P. Muraca, and P. Pugliese, "Extended and unscented kalman filters for mobile robot localization and environment reconstruction," in *21st Mediterranean Conference on Control and Automation*. IEEE, 2013, pp. 19–26.

[77] M. A. Skoglund, G. Hendeby, and D. Axehill, "Extended kalman filter modifications based on an optimization view point," in *18th International Conference on Information Fusion*. IEEE, 2015, pp. 1856–1861.

[78] J. Dunik, O. Straka, and M. Simandl, "On autocovariance least-squares method for noise covariance matrices estimation," *IEEE Transactions on Automatic Control*, vol. 62, no. 2, pp. 967–972, 2017.

[79] M. A. Zagrobelny and J. B. Rawlings, "Practical improvements to autocovariance least-squares," *AIChE Journal*, vol. 61, no. 6, pp. 1840–1855, 2015.

[80] Y. Meng, S. Gao, Y. Zhong, G. Hu, and A. Subic, "Covariance matching based adaptive unscented kalman filter for direct filtering in ins/gnss integration," *Acta Astronautica*, vol. 120, pp. 171–181, 2016.

[81] W. Zhang, W. Shi, and Z. Ma, "Adaptive unscented kalman filter based state of energy and power capability estimation approach for lithium-ion battery," *Journal of Power Sources*, vol. 289, pp. 50–62, 2015.

[82] M. A. Zagrobelny and J. B. Rawlings, "Identifying the uncertainty structure using maximum likelihood estimation," in *American Control Conference (ACC2015)*. IEEE, 2015, pp. 422–427.

[83] A. Aubry, A. De Maio, L. Pallotta, and A. Farina, "Maximum likelihood estimation of a structured covariance matrix with a condition number constraint," *IEEE Transactions on Signal Processing*, vol. 60, no. 6, pp. 3004–3021, 2012.

[84] Z. Weng and P. M. Djuric, "A bayesian approach to covariance estimation and data fusion," in *Proceedings of the 20th European Signal Processing Conference (EUSIPCO2012)*. IEEE, 2012, pp. 2352–2356.

[85] Y. Wang and P. M. Djuric, "Distributed bayesian estimation of linear models with unknown observation covariances," *IEEE Transactions on Signal Processing*, vol. 64, no. 8, pp. 1962–1971, 2016.

[86] S. Akhlaghi, N. Zhou, and Z. Huang, "Adaptive adjustment of noise covariance in kalman filter for dynamic state estimation," *arXiv preprint arXiv:1702.00884*, pp. 1–5, 2017.

[87] H. Wang, Z. Deng, B. Feng, H. Ma, and Y. Xia, "An adaptive kalman filter estimating process noise covariance," *Neurocomputing*, vol. 223, pp. 12–17, 2017.

[88] B. Feng, M. Fu, H. Ma, Y. Xia, and B. Wang, "Kalman filter with recursive covariance estimation—sequentially estimating process noise covariance," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 11, pp. 6253–6263, 2014.

[89] S. Thrun *et al.*, "Robotic mapping: A survey," *Exploring artificial intelligence in the new millennium*, vol. 1, pp. 1–35, 2002.

[90] T. Colleens and J. Colleens, "Occupancy grid mapping: An empirical evaluation," in *Mediterranean Conference on Control and Automation*. IEEE, 2007, pp. 1–6.

[91] G. Dissanayake, S. B. Williams, H. Durrant-Whyte, and T. Bailey, "Map management for efficient simultaneous localization and mapping (slam)," *Autonomous Robots*, vol. 12, no. 3, pp. 267–286, 2002.

[92] J. Levinson, M. Montemerlo, and S. Thrun, "Map-based precision vehicle localization in urban environments." in *Robotics: Science and Systems*, vol. 4. Citeseer, 2007, p. 1.

[93] H. Lategahn, A. Geiger, and B. Kitt, "Visual slam for autonomous ground vehicles," in *IEEE International Conference on Robotics and Automation (ICRA2011)*. IEEE, 2011, pp. 1732–1737.

[94] R. Kuramachi, A. Ohsato, Y. Sasaki, and H. Mizoguchi, "G-icp slam: An odometry-free 3d mapping system with robust 6dof pose estimation," in *IEEE International Conference on Robotics and Biomimetics (ROBIO2015)*. IEEE, 2015, pp. 17–181.

[95] R. C. Luo and C. C. Lai, "Multisensor fusion-based concurrent environment mapping and moving object detection for intelligent service robotics," *IEEE transactions on industrial electronics*, vol. 61, no. 8, pp. 4043–4051, 2014.

[96] F. M. Moreno, "Fusion multi-sensorial en mapas de ocupacion," Master's thesis, Universidad Carlos III de Madrid, 2016.

[97] A. Discant, A. Rogozan, C. Rusu, and A. Bensrhair, "Sensors for obstacle detection-a survey," in *International Spring Seminar on Electronics Technology*. IEEE, 2007, pp. 100–105.

[98] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1773–1795, 2013.

[99] A. Mukhtar, L. Xia, and T. B. Tang, "Vehicle detection techniques for collision avoidance systems: A review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2318–2338, 2015.

[100] C. Premebida, G. Monteiro, U. Nunes, and P. Peixoto, "A lidar and vision-based approach for pedestrian and vehicle detection and tracking," in *Intelligent Transportation Systems Conference (ITSC2007)*. IEEE, 2007, pp. 1044–1049.

[101] W. Zhang, "Lidar-based road and road-edge detection," in *Intelligent Vehicles Symposium (IV2010)*. IEEE, 2010, pp. 845–848.

[102] G. Prabhakar, B. Kailath, S. Natarajan, and R. Kumar, "Obstacle detection and classification using deep learning for tracking in high-speed autonomous driving," in *IEEE Region 10 Symposium (TENSYMP2017)*. IEEE, 2017, pp. 1–6.

[103] S. Scheidegger, J. Benjaminsson, E. Rosenberg, A. Krishnan, and K. Granstrom, "Mono-camera 3d multi-object tracking using deep learning detections and pmbm filtering," *arXiv preprint arXiv:1802.09975*, pp. 1–8, 2018.

[104] D. Gonzalez, J. Perez, V. Milanes, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.

[105] P. Bhattacharya and M. L. Gavrilova, "Voronoi diagram in optimal path planning," in *Voronoi Diagrams in Science and Engineering, 2007. ISVD'07. 4th International Symposium on*. IEEE, 2007, pp. 38–47.

[106] M. Seda, "Roadmap methods vs. cell decomposition in robot motion planning," in *Proceedings of the International Conference on Signal Processing, Robotics and Automation*. WSEAS, 2007, pp. 127–132.

[107] P. Vadakkepat, K. C. Tan, and W. Ming-Liang, "Evolutionary artificial potential fields and their application in real time robot path planning," in *IEEE Congress on Evolutionary Computation*, vol. 1. IEEE, 2000, pp. 256–263.

[108] T. Ju, S. Liu, J. Yang, and D. Sun, "Rapidly exploring random tree algorithm-based path planning for robot-aided optical manipulation of biological cells," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 3, pp. 649–657, 2014.

[109] C. Luo, S. X. Yang, X. Li, and M. Q.-H. Meng, "Neural-dynamics-driven complete area coverage navigation through cooperation of multiple mobile robots," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 1, pp. 750–760, 2017.

[110] P. Bender, O. S. Tas, J. Ziegler, and C. Stiller, "The combinatorial aspect of motion planning: Maneuver variants in structured environments," *IEEE Intelligent Vehicles Symposium (IV)*, pp. 1386–1392, 2015.

[111] X. Qian, F. Altche, P. Bender, C. Stiller, and A. de La Fortelle, "Optimal trajectory planning for autonomous driving integrating logical constraints: An miqp perspective," *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pp. 205–210, 2016.

[112] C. Englund, L. Chen, J. Ploeg, E. Semsar-Kazerooni, A. Voronov, H. H. Bengtsson, and J. Didoff, "The grand cooperative driving challenge 2016: boosting the introduction of cooperative automated vehicles," *IEEE Wireless Communications*, vol. 23, no. 4, pp. 146–152, 2016.

[113] H. Rewald and O. Stursberg, "Cooperation of autonomous vehicles using a hierarchy of auction-based and model-predictive control," *IEEE Intelligent Vehicles Symposium (IV)*, pp. 1078–1084, 2016.

[114] M. Naumann and C. Stiller, "Towards cooperative motion planning for automated vehicles in mixed traffic," *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–6, 2017.

[115] S. Manzinger, M. Leibold, and M. Althoff, "Driving strategy selection for cooperative vehicles using maneuver templates," *IEEE Intelligent Vehicles Symposium (IV)*, pp. 647–654, 2017.

[116] J. Luo and J.-P. Hubaux, "A survey of inter-vehicle communication," *Infoscience - LCA-REPORT-2004-013*, pp. 1–12, 2004.

[117] S. K. Gehrig and F. J. Stein, "Collision avoidance for vehicle-following systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 2, pp. 233–244, 2007.

[118] S. E. Shladover, "Automated vehicles for highway operations (automated highway systems)," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 219, no. 1, pp. 53–75, 2005.

[119] S. E. Shladover, "Cooperative (rather than autonomous) vehicle-highway automation systems," *IEEE Intelligent Transportation Systems Magazine*, vol. 1, no. 1, pp. 10–19, 2009.

[120] P. Ioannou, *Automated highway systems*. Springer Science and Business Media, 2013.

[121] S. Meena and O. Prakash, "The study on automated highway systems," *Imperial Journal of Interdisciplinary Research*, vol. 3, no. 4, 2017.

[122] B. Gerkey and J. Mataric, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[123] L. E. Parker, *Multiple Mobile Robot Systems*. Springer, 2008.

[124] M. Badreldin, A. Hussein, and A. Khamis, "A comparative study between optimization and market-based approaches to multi-robot task allocation," *Journal of Advances in Artificial Intelligence*, vol. 2013, pp. 1–11, 2013.

[125] A. Ezz, "Adaptive optimal task assignment for cooperative autonomous vehicles," Master's thesis, German University in Cairo, 2018.

[126] N. Atay and B. Bayazit, "Mixed-integer linear programming solution to multi-robot task allocation problem," *Washington Univ., St. Louis, Tech. Rep. WUCSE-2006-54*, vol. 54, 2006.

[127] A. R. Mosteo, "Multi-robot task allocation for service robotics: From unlimited to limited communication range," Ph.D. dissertation, Universidad de Zaragoza, 2010.

[128] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1257–1270, 2006.

[129] A. Hussein, "A market-based approach to multi-robot task allocation problem," Master's thesis, German University in Cairo, 2013.

[130] W. Kmiecik, M. Wojcikowski, L. Koszalka, and A. Kasprzak, *Task Allocation in Mesh Connected Processors with Local Search Meta-heuristic Algorithms*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, vol. 5991.

[131] W. Chen and C. Lin, "A hybrid heuristic to solve a task allocation problem," *Computers & Operations Research*, vol. 27, no. 3, pp. 287–303, 2000.

[132] C. Liu and A. Kroll, *A Centralized Multi-Robot Task Allocation for Industrial Plant Inspection by Using A* and Genetic Algorithms*, ser. Lecture Notes in Computer

Science, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. Zadeh, and J. Zurada, Eds. Springer Berlin Heidelberg, 2012, vol. 7268.

[133] R. Luna and K. E. Bekris, "Efficient and complete centralized multi-robot path planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2011)*. IEEE, 2011, pp. 3268–3275.

[134] E. G. Hernandez-Martinez and E. Aranda-Bricaire, "Decentralized formation control of multi-agent robot systems based on formation graphs," *Studies in Informatics and Control*, vol. 21, no. 1, pp. 7–16, 2012.

[135] P. Yang, R. A. Freeman, G. J. Gordon, K. M. Lynch, S. S. Srinivasa, and R. Sukthankar, "Decentralized estimation and control of graph connectivity for mobile sensor networks," *Automatica*, vol. 46, no. 2, pp. 390–396, 2010.

[136] Z. Jian, P. Zhihong, and L. Bo, "Multi-task allocation of ucavs considering time cost and hard time window constraints," *IEEE 31st Chinese Control Conference (CCC)*, pp. 2448–2452, 2012.

[137] A. Khamis, A. Hussein, and A. Elmogy, *Cooperative Robots and Sensor Networks*. Springer International Publishing, 2015, ch. Multi-robot Task Allocation: A Review of the State-of-the-Art, pp. 31–51.

[138] T. Banziger, A. Kunz, and K. Wegener, "Optimizing human–robot task allocation using a simulation tool based on standardized work descriptions," *Journal of Intelligent Manufacturing*, pp. 1–14, 2018.

[139] A. Hussein, A. Diaz-Alvarez, J. M. Armingol, and C. Olaverri-Monreal, "3dcoautosim: Simulator for cooperative adas and automated vehicles," in *IEEE International Conference on Intelligent Transportation Systems (ITSC2018)*. IEEE, 2018, pp. 1–6, under-review.

[140] A. Hussein, F. Garcia, and C. Olaverri-Monreal, *Hands-on Tutorial: How to use Unity as Simulator for ROS-based Applications*. Springer International Publishing, 2018, vol. 4, ch. ROS-Based Approach for Unmanned Vehicles in Civil Applications, pp. 1–22, under-review.

[141] B. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proceedings of the 11th international conference on advanced robotics*, vol. 1, 2003, pp. 317–323.

[142] O. S. R. Foundation, "Gazebo," [Last Accessed: 2018-04-30]. [Online]. Available: https://bitbucket.org/osrf/gazebo

[143] O. S. R. Foundation, "Car demo," [Last Accessed: 2018-04-30]. [Online]. Available: https://github.com/osrf/car_demo

[144] A. S. Kyaw, *Unity 4.x Game AI Programming*. Packt Publishing Ltd, 2013.

[145] S. Christodoulou, D. Michael, A. Gregoriades, and M. Pampaka, "Design of a 3d interactive simulator for driver behavior analysis," in *Proceedings of the 2013 Summer Computer Simulation Conference*. Society for Modeling & Simulation International, 2013, p. 17.

[146] A. Smid, "Comparison of unity and unreal engine," Ph.D. dissertation, Czech Technical University in Prague, 2017.

[147] C. Biurrun, L. Serrano-Arriezu, and C. Olaverri-Monreal, "Micro-scopic driver-centric simulator: Linking unity3d and sumo," in *World Conference on Information Systems and Technologies*. Springer, 2017, pp. 851–860.

[148] C. Olaverri-Monreal, J. Errea-Moreno, and A. Diaz-Alvarez, "Implementation and evaluation of a traffic light assistance system in a simulation framework based on v2i communication," *Journal of Advanced Transportation*, 2018.

[149] F. Michaeler and C. Olaverri-Monreal, "3d driving simulator with vanet capabilities to assess cooperative systems: 3dsimvanet," in *Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 999–1004.

[150] S. P. Singh, K. Jain, and V. R. Mandla, "Image based virtual 3d campus modeling by using cityengine," *American Journal of Engineering Science and Technology Research*, vol. 2, no. 1, pp. 1–10, 2014.

[151] B. Ozdodanlar, "Realistic car controller v3.1," *Bone Cracker Games*, vol. 3, pp. 1–36, 2017.

[152] P. Marin-Plaza, A. Hussein, D. Martin, and A. d. l. Escalera, "Global and local path planning study in a ros-based research platform for autonomous vehicles," *Journal of Advanced Transportation*, vol. 2018, pp. 1–11, 2018.

[153] M. S. Aminian, A. Allamehzadeh, M. Mostaed, and C. Olaverri-Monreal, "Cost-efficient traffic sign detection relying on smart mobile devices," in *International Conference on Computer Aided Systems Theory*. Springer, 2017, pp. 419–426.

[154] A. Allamehzadeh, J. U. de la Parra, A. Hussein, F. Garcia, and C. Olaverri-Monreal, "Cost-efficient driver state and road conditions monitoring system for conditional automation," in *IEEE Intelligent Vehicles Symposium (IV2017)*. IEEE, 2017, pp. 1497–1502.

[155] C. Olaverri-Monreal, R. Lorenz, F. Michaeler, G. C. Krizek, and M. Pichler, "Tailigator: Cooperative system for safety distance observance," in *International Conference on Collaboration Technologies and Systems (CTS)*. IEEE, 2016, pp. 392–397.

[156] C. Olaverri-Monreal, G. C. Krizek, F. Michaeler, R. Lorenz, and M. Pichler, "Collaborative approach for a safe driving distance using stereoscopic image processing," *Future Generation Computer Systems*, 2018.

[157] K. Abdelgawad, J. Gausemeier, R. Dumitrescu, M. Grafe, J. Stoecklein, and J. Berssenbruegge, "Networked driving simulation: Applications, state of the art, and design considerations," *Designs*, vol. 2017, pp. 1–17, 2017.

[158] A. Hussein, A. Al-Kaff, A. de la Escalera, and J. M. Armingol, "Autonomous indoor navigation of low-cost quadcopters," in *IEEE International Conference on Service Operations And Logistics, And Informatics (SOLI2015)*. IEEE, 2015, pp. 133–138.

[159] D. Martin, P. M. Plaza, A. Hussein, A. de la Escalera, and J. M. Armingol, *UNED Plasencia Revista de Investigacion Universitaria*. Agbatanero, 2016, vol. 16, ch. Ros-based architecture for autonomous intelligent campus automobile (icab), pp. 257–272.

[160] P. Marin-Plaza, J. Beltran, A. Hussein, B. Musleh, D. Martin, A. de la Escalera, and J. M. Armingol, "Stereo vision-based local occupancy grid map for autonomous navigation in ros," in *Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP2016)*, vol. 3. SciTePress, 2016, pp. 703–708.

[161] A. Hussein, P. Marin-Plaza, D. Martin, A. de la Escalera, and J. M. Armingol, "Autonomous off-road navigation using stereo-vision and laser-rangefinder fusion for outdoor obstacles detection," in *IEEE Intelligent Vehicles Symposium (IV2016)*. IEEE, 2016, pp. 104–109.

[162] P. M. Plaza, A. Hussein, C. Guindel, F. Garcia, D. Martin, and A. de la Escalera, "Arquitectura basada en ros para el vehiculo icab (intelligent campus automobile)," in *XXXVII Jornadas de Automatica*, vol. 37. UNED, 2016, pp. 1–6.

[163] P. Marin-Plaza, A. Hussein, D. Martin, and A. de la Escalera, "Complete ros-based architecture for intelligent vehicles," *Iberian Robotics (ROBOT2017)*, vol. 2017, pp. 499–510, 2017.

[164] N. Hernandez, A. Hussein, D. Cruzado, I. Parra, and J. M. Armingol, "Applying low cost wifi-based localization to in-campus autonomous vehicles," in *IEEE International Conference on Intelligent Transportation Systems (ITSC2017)*. IEEE, 2017, pp. 1–6.

[165] A. Astudillo, F. M. Moreno, A. Hussein, and F. Garcia, "Cost-efficient brainwave controller for automated vehicles route decisions," in *IEEE International Conference on Intelligent Transportation Systems (ITSC2017)*. IEEE, 2017, pp. 51–56.

[166] M. Osman, A. Hussein, A. Al-Kaff, F. Garcia, and J. M. Armingol, "Online adaptive covariance estimation approach for multiple odometry sensors fusion," in *IEEE Intelligent Vehicles Symposium (IV2018)*. IEEE, 2018, pp. 1–6.

[167] A. Al-Kaff, F. M. Moreno, and A. Hussein, *Robot Operating System (ROS)*. Springer International Publishing, 2018, vol. 3, ch. ROS-Based Approach for Unmanned Vehicles in Civil Applications, pp. 1–30.

[168] M. Osman, R. Alonso, F. M. Moreno, A. Hussein, and A. Al-Kaff, "Extended h-infinity filter for multi-sensor fusion localization," in *IEEE International Conference on Intelligent Transportation Systems (ITSC2018)*. IEEE, 2018, pp. 1–6.

[169] A. H. Abdelhamid, M. A. de Miguel, P. M. Plaza, A. Hussein, and F. Garcia, "Model predictive linear quadratic regulator based path tracking for automated ground vehicles," in *IEEE International Conference on Intelligent Transportation Systems (ITSC2018)*. IEEE, 2018, pp. 1–6, under-review.

[170] A. Al-Kaff, R. Alonso, M. Osman, and A. Hussein, "Skyonyx: Autonomous uav research platform for air transportation system (atsys)," in *IEEE International Conference on Intelligent Transportation Systems (ITSC2018)*. IEEE, 2018, pp. 1–6, under-review.

[171] P. Marin-Plaza, A. Hussein, D. Martin, and A. de la Escalera, "icab use case for ros-based architecture," *Iberian Robotics (ROBOT2018)*, vol. 2018, pp. 1–31, 2018, under-review.

[172] E. Upton, "Raspberry pi 3," [Last Accessed: 2018-04-30]. [Online]. Available: https://www.raspberrypi.org/blog/raspberry-pi-3-on-sale

[173] S. Korkalainen, "Turtlebot3-robotit," *Metropolia Ammattikorkeakoulu*, pp. 1–40, 2017.

[174] E. Guizzo and E. Ackerman, "The turtlebot3 teacher [resources hands on]," *IEEE Spectrum*, vol. 54, no. 8, pp. 19–20, 2017.

[175] V. Rabaud *et al.*, "Slam gmapping," [Last Accessed: 2018-04-30]. [Online]. Available: https://github.com/ros-perception/slam_gmapping

[176] E. Marder-Eppstein, D. V. Lu, M. Ferguson, A. Hoy *et al.*, "Ros navigation stack," [Last Accessed: 2018-04-30]. [Online]. Available: https://github.com/ros-planning/navigation

[177] Robotis, "Turtlebot3 emanual," [Last Accessed: 2018-04-30]. [Online]. Available: http://emanual.robotis.com/docs/en/platform/turtlebot3/overview

[178] A. Al-Kaff, "Vision-based navigation system for unmanned aerial vehicles," Ph.D. dissertation, Universidad Carlos III de Madrid, 2017.

[179] A. Al-Kaff, A. de La Escalera, and J. M. Armingol, "Homography-based navigation system for unmanned aerial vehicles," in *International Conference on Advanced Concepts for Intelligent Vision Systems*. Springer, 2017, pp. 288–300.

[180] J. T. Isaacs, C. Magee, A. Subbaraman, F. Quitin, K. Fregene, A. R. Teel, U. Madhow, and J. P. Hespanha, "Gps-optimal micro air vehicle navigation in degraded environments," in *American Control Conference (ACC)*. IEEE, 2014, pp. 1864–1871.

[181] M. F. Bahat and T. Filik, "Gps-based antenna tracking and signal beamforming system for small uav platform," in *Signal Processing and Communications Applications Conference (SIU)*. IEEE, 2015, pp. 1977–1980.

[182] A. Al-Kaff, F. Garcia, D. Martin, A. De La Escalera, and J. M. Armingol, "Obstacle detection and avoidance system based on monocular camera and size expansion algorithm for uavs," *Sensors*, vol. 17, no. 5, p. 1061, 2017.

[183] A. Al-Kaff, F. M. Moreno, A. de la Escalera, and J. M. Armingol, "Intelligent vehicle for search, rescue and transportation purposes," in *International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, 2017, pp. 110–115.

[184] R. Rao, "Steering linkage design. a method of determining the configuration of the steering linkage so that the geometry conforms to ackermann principle," *Automotive Engineering*, vol. 58, pp. 31–33, 1968.

[185] E. Mohamed and S. Albatlan, "Modeling and experimental design approach for integration of conventional power steering and a steer-by-wire system based on active steering angle control," *American Journal of Vehicle Design*, vol. 2, no. 1, pp. 32–42, 2014.

[186] D. Garcia, "Interfaz de control icab, el vehiculo autonomo," Bachelor Thesis, Universidad Carlos III de Madrid, 2015.

[187] Y. Lertxundi, "Steering controller for intelligent vehicle," Bachelor Thesis, Universidad Carlos III de Madrid, 2017.

[188] E. Hisham, "Modeling and control of steer by wire system," ASU Bachelor Thesis Abroad, Universidad Carlos III de Madrid, 2018.

[189] A. Nasr, "Weight adaptive braking control for autonomous ground vehicles," ASU Bachelor Thesis Abroad, Universidad Carlos III de Madrid, 2018.

[190] P. M. Plaza, "icab ros architecutre," Ph.D. dissertation, Universidad Carlos III de Madrid, 2018.

[191] S. H. Juan and F. H. Cotarelo, "Multi-master ros systems," *Institut de Robotics and Industrial Informatics*, pp. 1–18, 2015.

[192] R. Peter Bonasso, R. James Firby, E. Gat, D. Kortenkamp, D. P. Miller, and M. G. Slack, "Experiences with an architecture for intelligent, reactive agents," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 9, no. 2-3, pp. 237–256, 1997.

[193] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "The pixhawk open-source computer vision framework for mavs," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 38, no. 1, p. C22, 2011.

[194] A. Majdik, M. Popa, L. Tamas, I. Szoke, and G. Lazea, "New approach in solving the kidnapped robot problem," in *Joint conference of the 41st International Symposium on Robotics (ISR2010) and the 6th German Conference on Robotics (ROBOTIK2010)*. VDE, 2010, pp. 1–6.

[195] Z. Su, X. Zhou, T. Cheng, H. Zhang, B. Xu, and W. Chen, "Global localization of a mobile robot using lidar and visual features," in *IEEE International Conference on Robotics and Biomimetics (ROBIO2017)*. IEEE, 2017, pp. 2377–2383.

[196] C. N. Taylor and D. Mohler, "A study of particle filtering approaches for the kidnapped robot problem," in *Signal Processing, Sensor/Information Fusion, and Target Recognition XXVII*, vol. 10646. International Society for Optics and Photonics, 2018, p. 106460A.

[197] B. Zhang, J. Liu, and H. Chen, "Amcl based map fusion for multi-robot slam with heterogenous sensors," in *International Conference on Information and Automation*, 2013, pp. 1–6.

[198] D. Cruzado, "Laser multiplano aplicado a los vehiculos autonomos: Mapeado y localizacion inicial," Master's thesis, Universidad Carlos III de Madrid, 2017.

[199] B. Musleh, D. Martin, A. de la Escalera, and J. M. Armingol, "Visual ego motion estimation in urban environments based on uv disparity," in *IEEE Intelligent Vehicles Symposium (IV2012)*. IEEE, 2012, pp. 444–449.

[200] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time." in *Robotics: Science and Systems*, vol. 2, 2014, pp. 1–9.

[201] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[202] A. Rudolph, "Quantification and estimation of differential odometry errors in mobile robotics with redundant sensor information," *The International Journal of Robotics Research*, vol. 22, no. 2, pp. 117–128, 2003.

[203] K. Lee, W. Chung, and K. Yoo, "Kinematic parameter calibration of a car-like mobile robot to improve odometry accuracy," *Mechatronics*, vol. 20, no. 5, pp. 582–595, 2010.

[204] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium*. IEEE, 2000, pp. 153–158.

[205] T. Moore and D. Stouch, "A generalized extended kalman filter implementation for the robot operating system," in *Intelligent Autonomous Systems 13*. Springer, 2016, pp. 335–348.

[206] A. J. Rincon, "Mapeado con camara tof para vehiculos inteligentes," Master's thesis, Universidad Carlos III de Madrid, 2018.

[207] J. Beltran, C. Jaraquemada, B. Musleh, A. de la Escalera, and J. M. Armingol, "Dense semantic stereo labelling architecture for in-campus navigation," in *Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP2017)*, 2017, pp. 266–273.

[208] C. Rosmann, F. Hoffmann, and T. Bertram, "Timed-elastic-bands for time-optimal point-to-point nonlinear model predictive control," in *European Control Conference (ECC2015)*. IEEE, 2015, pp. 3352–3357.

[209] K. Nam, S. Oh, H. Fujimoto, and Y. Hori, "Estimation of sideslip and roll angles of electric vehicles using lateral tire force sensors through rls and kalman filter approaches," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 3, pp. 988–1000, 2013.

[210] J. Y. Wong, *Theory of ground vehicles*. John Wiley and Sons, 2008.

[211] A. Hussein, F. Garcia, J. M. Armingol, and C. Olaverri-Monreal, "P2v and v2p communication for pedestrian warning on the basis of autonomous vehicles," in *IEEE International Conference on Intelligent Transportation Systems (ITSC2016)*. IEEE, 2016, pp. 2034–2039.

[212] A. Kokuti, A. Hussein, P. M. Plaza, A. de la Escalera, and F. Garcia, "V2x communications architecture for off-road autonomous vehicles," in *IEEE International Conference on Vehicular Electronics and Safety (ICVES2017)*. IEEE, 2017, pp. 69–74.

[213] A. Hussein, P. Marin-Plaza, F. Garcia, and J. M. Armingol, "Autonomous cooperative driving using v2x communications in off-road environment," in *IEEE International Conference on Intelligent Transportation Systems (ITSC2017)*. IEEE, 2017, pp. 1–6.

[214] J. Harding, G. Powell, R. Yoon, J. Fikentscher, C. Doyle, D. Sade, M. Lukuc, J. Simons, and J. Wang, "Vehicle-to-vehicle communications: Readiness of v2v technology for application," *USDOT NHTSA*, pp. 1–327, 2014.

[215] S. Biswas, R. Tatchikou, and F. Dion, "Vehicle-to-vehicle wireless communication protocols for enhancing highway traffic safety," *IEEE Communications Magazine*, vol. 44, no. 1, pp. 74–82, 2006.

[216] L. Tellis, F. Ahmed-Zaid, J. E. Stinnett, C. Nave, T. E. Pilutti, T. D. Zwicky, J. A. Martell, and J. C. Ivan, "Vehicle-to-vehicle/vehicle-to-infrastructure control," *IEEE The Impact of Control Technology*, 2011.

[217] M. Bagheri, M. Siekkinen, and J. K. Nurminen, "Cellular-based vehicle to pedestrian (v2p) adaptive communication for collision avoidance," in *IEEE International Conference on Connected Vehicles and Expo (ICCVE2014)*. IEEE, 2014, pp. 450–456.

[218] F. Garcia, D. Martin, A. de la Escalera, and J. M. Armingol, "In-vehicle sensor fusion methodology for pedestrian detection with danger estimation," in *Congreso Espanol sobre Tecnologias y Ligica Fuzzy (ESTYLF2014)*, 2014.

[219] G. Johansson and K. Rumar, "Drivers' brake reaction times," *Human factors*, vol. 13, no. 1, pp. 23–27, 1971.

[220] H. Makishita and K. Matsunaga, "Differences of drivers reaction times according to age and mental workload," *Accident Analysis and Prevention*, vol. 40, no. 2, pp. 567–575, 2008.

[221] G. A. Association *et al.*, "The case for cellular v2x for safety and cooperative driving," *5GAA Whitepaper*, 2016.

[222] E. Ndashimye, N. I. Sarkar, and S. K. Ray, "A novel network selection mechanism for vehicle-to-infrastructure communication," *IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC)*, pp. 483–488, 2016.

[223] A. Kokuti, A. Hussein, A. de la Escalera, and F. Garcia, "Market-based approach for cooperation and coordination among multiple autonomous vehicles," in *IEEE International Conference on Intelligent Transportation Systems (ITSC2017)*. IEEE, 2017, pp. 534–539.

[224] A. Hussein, P. M. Plaza, F. Garcia, and J. M. Armingol, "Optimization-based approach for cooperation and coordination of multi-autonomous vehicles," in *Sixteenth International Conference on Computer Aided Systems Theory (EUROCAST)*. Springer International Publishing, 2017, pp. 1–2.

[225] A. Hussein, P. Marin-Plaza, F. Garcia, and J. M. Armingol, "Hybrid optimization-based approach for multiple intelligent vehicles requests allocation," *Journal of Advanced Transportation*, vol. 2018, pp. 1–12, 2018.

[226] A. Hussein, F. M. Moreno, F. Garcia, and J. M. Armingol, "Multiple vehicle cooperation generic ros-based architecture for transportation requests," *Integrated Computer-Aided Engineering*, vol. 2018, pp. 1–16, 2018.

[227] A. Maji and M. K. Jha, "Multi-objective highway alignment optimization using a genetic algorithm," *Journal of Advanced Transportation*, vol. 43, no. 4, pp. 481–504, 2009.

[228] A. Lewandowski, S. Bocker, V. Koster, and C. Wietfeld, "Design and performance analysis of an ieee 802.15. 4 v2p pedestrian protection system," in *IEEE International Symposium on Wireless Vehicular Communications (WiVeC2013)*. IEEE, 2013, pp. 1–6.

[229] R. Necula, M. Breaban, and M. Raschip, "Tackling the bi-criteria facet of multiple traveling salesman problem with ant colony systems," *IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 873–880, 2015.

[230] R. Necula, M. Breaban, and M. Raschip, "Performance evaluation of ant colony systems for the single-depot multiple traveling salesman problem," *International Conference on Hybrid Artificial Intelligence Systems*, pp. 257–268, 2015.